



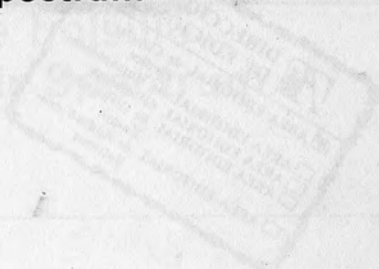
Manual de bolsillo del

SINCLAIR SPECTRUM



S. VICKERS

Manual de bolsillo del Sinclair Spectrum



Steven Vickers

Manual de bolsillo del Sinclair Spectrum

Steven Vickers

Versión española:

Miguel Angel Fernández Alvarez

 **Alhambra**

Edición original:
POCKET GUIDE. SINCLAIR SPECTRUM

© Pitman Publishing LTD, 1984

Primera edición española, 1985
Reimpresión, 1985

© EDITORIAL ALHAMBRA, S.A.
R.E. 182
28001 Madrid. Claudio Coello, 76

Delegaciones:

08008 Barcelona. Enrique Granados, 61
48014 Bilbao. Iruña, 12
18009 Granada. Pza. de las Descalzas, 2
15005 La Coruña. Pasadizo de Pernas, 13
28002 Madrid. Saturnino Calleja, 1
33006 Oviedo. Avda. del Cristo, 9
38004 Santa Cruz de Tenerife. General Portier, 14
41012 Sevilla. Reina Mercedes, 35
46003 Valencia. Cabillers, 5
47014 Valladolid. Gavilla, 3
50005 Zaragoza. Concepción Arenal, 25

México

Editorial Alhambra Mexicana, S.A.
Avda. División del Norte, 2412
03340 México, D.F.

Argentina

EDICLE
Juncal, 4649/51
1425 Buenos Aires

Colombia

Editorial Alhambra Colombiana, Ltda.
Calle 117 n° 11, A-65
Bogotá

n c 06120040

ISBN 84-205-1080-7

ISBN 0-273-02075-7, Pitman Publishing LTD.
Londres, edición original

Depósito legal: M. 33.540—1985

Reservados todos los derechos. Ni la totalidad, ni parte de este libro pueden reproducirse o transmitirse, utilizando medios electrónicos o mecánicos, por fotocopia, grabación, información, anulado, u otro sistema, sin permiso por escrito del editor.

Cubierta: Estudio Enlace
Fotocomposición: SAF
Impresión: Artes Gráficas Benzal, S.A.
Papel: Kanguros
Encuadernación: Gómez Pinto, S.A.

Impreso en España - Printed in Spain

Indice general

1.	Cómo utilizar este Manual de Bolsillo	1
2.	El ordenador ZX Spectrum	2
3.	Proceso de programas	4
4.	Glosario de palabras BASIC	11
5.	Expresiones	34
6.	Estructuras	47
7.	Color y gráficos	50
8.	Impresión en la pantalla de TV o en la impresora ZX, INPUT	56
9.	Almacenamiento en cinta cassette	64
10.	Entrada y salida en general	67
11.	El Interface 1 ZX	74
12.	Mensajes informativos	80
	Indice de materias	85

1. Cómo utilizar este Manual de Bolsillo

1.1. Para qué sirve y a quién va dirigido

No es nuestra intención sustituir el manual de instrucciones suministrado por Sinclair, sino facilitar una referencia rápida. Hemos partido de la suposición de que el lector posee, al menos, un conocimiento somero del lenguaje BASIC, y de que necesita conocer rápidamente la respuesta a cuestiones específicas acerca de la programación del ZX Spectrum. Si no ha llegado Vd. a este punto, le serán de utilidad otros dos manuales de bolsillo Pitman, concretamente los dedicados a la programación en general y al lenguaje BASIC.

1.2. Cómo encontrar el tema deseado

Si desea averiguar qué tipo de ordenador es el Spectrum, lea la Sección 2.

Si no está lo suficientemente familiarizado con el Spectrum y tiene dificultades para hacer funcionar sus programas, consulte el apartado 2.1.

En la Sección 3 se estudian las generalidades del funcionamiento del Spectrum.

Si lo que quiere es averiguar detalles acerca de una característica en particular, puede buscar la correspondiente palabra clave de BASIC en el glosario —Sección 4— donde encontrará referencias a otras partes del manual que guardan relación con el tema, o bien buscar el apartado pertinente en el índice.

En el caso de que le haya desorientado un mensaje de error que no acaba de entender, pruebe a leer la Sección 12 y después consulte la sentencia o función BASIC correspondiente en el glosario.

1.3. Notación

Cuando se describe una característica del lenguaje BASIC es preciso especificar su formato o *sintaxis*. Emplearemos letras mayúsculas y signos de puntuación para representar la parte obligatoria de la sentencia, y letras minúsculas para indicaciones de carácter general relativas a los elementos que hay que incluir para completarla. Por ejemplo, la *sintaxis* de una sentencia LET es:

LET variable = expresión

donde 'variable' y 'expresión' representan a cualquier variable o expresión con las que se desee trabajar.

Si bien una expresión puede adoptar un formato general cualquiera (véase la Sección 5), un *argumento* es la forma más restringida que puede utilizarse como variable independiente ("argumento") de una función; por ejemplo, la sintaxis de la función SIN (seno) es la siguiente:

SIN argumento numérico

Entraremos en detalle sobre esto en el apartado 5.6; en la práctica significa que la mayoría de las expresiones deben ir entre paréntesis para ser consideradas como argumentos (por ejemplo, SIN (x + y)), mientras que otras más sencillas, tales como números, cadenas y variables no les necesitan (por ejemplo, SIN 3,9 o SIN x).

Una expresión o argumento *entero* son valores numéricos redondeados al entero (número sin cifras decimales) más próximo antes de su utilización. Por ejemplo, la sintaxis de la sentencia PLOT (trazar) sería:

PLOT expresión entera, expresión entera

En el texto emplearemos comillas simples (') para encerrar un elemento de BASIC, tal como 'PRINT x', pero nunca forman parte de la expresión BASIC en sí misma.

2. El ordenador ZX Spectrum

El ordenador ZX Spectrum puede considerarse como la típica máquina barata con capacidad para colores y funcionamiento con BASIC. Se alimenta de la red eléctrica doméstica, y emplea un receptor normal de TV para presentar sus resultados y gráficos.

La presentación visual en la pantalla está compuesta por una retícula de alta resolución (256×192 pixels o puntos de imagen) y con un máximo de 15 colores simultáneamente, aunque existen limitaciones en cuanto a la proximidad de ciertos colores. También es posible hacer destellar intermitentemente algunas partes de la imagen.

Se pueden conservar programas de manera permanente en cintas cassette ordinarias (en caso contrario se pierden los programas al apagar el aparato) o utilizando el Microdrive Sinclair, un dispositivo más veloz y mejor organizado. Para emplear el Microdrive es necesario adquirir un Interface 1 ZX, mediante el cual es posible conectar el Spectrum a Microdrives, a otros ordenadores Spectrum, y en general, a cualquier aparato que funcione con un enlace RS232.

El ordenador lleva incorporado un altavoz bastante libre de ruidos con una sola voz.

El ordenador Spectrum funciona con el lenguaje de programación especial para principiantes BASIC, que a pesar de ser muy engorroso con programas largos, facilita un acceso sencillo y barato a las posibilidades de los ordenadores. Con sus 16K de memoria ROM pone en funcionamiento un dialecto muy parecido al BASIC convencional con ciertas características diseñadas especialmente para el Spectrum. Este dialecto resulta muy cómodo para producir gráficos y trabajar con almacenamiento por medio de cassettes, y en general, facilita el empleo de las características propias del Spectrum.

Existen dos modelos, uno de ellos con 16K de memoria RAM y otro con 48K. La presentación en pantalla ocupa siempre 6,75K de RAM. Gran parte de los mejores programas que hay a la venta no funcionan en el modelo de 16K.

Sinclair fabrica también una impresora barata de baja calidad para el Spectrum y el Interface 2 ZX, necesario para ejecutar programas comercializados en forma de cartuchos ROM, y que contiene un interface para palanca de r tula (*joystick*).

2.1. Idiosincrasias del BASIC Spectrum

Si ya tiene conocimiento del lenguaje BASIC pero se le plantean dificultades a la hora de hacer funcionar sus programas en el Spectrum, consulte los puntos siguientes:

Entrada por tecla  nica. Las palabras clave del lenguaje BASIC no se teclean letra por letra en el Spectrum, sino que existe un sistema de cambio de funciones de las teclas (*shift*) gracias al cual se pueden introducir todas las palabras claves con una sola pulsaci n. Adem s, al principio de una sentencia, el ordenador considera que la tecla pulsada corresponde a una palabra clave,

y no a una letra. Este sistema se aplica también a símbolos dobles, tales como ' $< =$ '. Consulte los apartados 3.2 y 3.4.

Errores de sintaxis. El sistema BASIC no aceptará sentencias cuyo formato (sintaxis) sea inválido; en tal situación, no pasará de la zona de escritura, mostrando un signo de interrogación intermitente, '?', en la posición inmediatamente anterior al primer error.

LET. No puede omitirse esta palabra clave.

IF... THEN. (SI... ENTONCES) Después de THEN no es válido escribir únicamente un número de línea, como en otras versiones de BASIC. Por ejemplo, hay que sustituir 'IF a = 2 THEN 150' por 'IF a = 2 THEN GO TO 150'.

LEFT\$, RIGHT\$ y MID\$. No existen estas funciones en la versión de BASIC Spectrum. En lugar de ellas se utiliza un método más flexible (fragmentación) para especificar partes de una cadena. Véase el apartado 5.5.

Operadores lógicos. Los operadores booleanos AND ("Y"), OR ("O") y NOT ("NO"), no se aplican a bits —véase el apartado 5.7—. Se comportan en la forma habitual con las comparaciones, =, <, etc.

Subíndices de matrices. Empiezan por 1, y no por 0.

3. Proceso de programas

3.1. División de la pantalla

La pantalla de TV está dividida funcionalmente en dos partes, estando reservada la inferior para escritura provisional.

La parte superior de la pantalla, que es la parte principal, se utiliza para el resultado de sentencias tales como PRINT y LIST. Cuando se ha ocupado toda la pantalla y se llega a la parte inferior, se produce un desplazamiento hacia arriba (*scrolling*) de toda la imagen, creando espacio para otra línea y haciendo desaparecer la línea superior.

La parte inferior de la pantalla sirve para teclear la entrada (ya se trate de líneas de instrucciones BASIC o de datos

correspondientes a sentencias INPUT) y también para presentar mensajes informativos. Normalmente ocupa la línea inferior dejando encima de ella una línea en blanco, pero si el texto abarca más de una línea, se amplía la parte inferior para incluirlo en su totalidad. De esta forma el ordenador puede manejar hasta 22 líneas de texto y una línea en blanco.

En la Sección 8 se estudia con más detalle la forma de utilizar ambas partes de la pantalla.

Cuando se modifica un programa aparece automáticamente un listado en la parte superior de la pantalla, listado que también puede obtenerse pulsando ENTER sin haber tecleado ninguna línea de BASIC.

La sucesión general de acontecimientos es la siguiente:

1. Se tecléa una línea de BASIC en la parte inferior de la pantalla y se pulsa ENTER a continuación.
2. La máquina comprueba la sintaxis; si no es correcta, hace aparecer intermitentemente un signo de interrogación '?' antes del primer error y espera a que el usuario lo corrija ('editando' la línea).
3. Si se trata de una línea de programa (distinguida por su número de línea), el ordenador la inserta en el programa, produce un listado automáticamente (en la parte superior de la pantalla) y espera a que el usuario teclee la línea siguiente.
4. En caso de ser una línea de comando, el ordenador la ejecuta de modo inmediato escribiendo el correspondiente resultado en la parte superior de la pantalla. Observe que el comando RUN (ejecución) lo primero que hace es borrar la pantalla, mientras que la mayoría de las otras sentencias no borran los resultados previos.
5. El ordenador exhibe el mensaje 'OK' —o cualquier otro mensaje pertinente— en la parte inferior de la pantalla y aguarda nuevas instrucciones (el mensaje desaparece al pulsar la primera tecla).

3.2. Introducción de texto - El cursor

Cuando se está escribiendo texto hay una letra intermitente, denominada *cursor*, en la parte inferior de la pantalla, cuya

posición indica el lugar donde se insertará el próximo carácter teclado —justo antes del cursor, desplazando horizontalmente todos los demás caracteres para hacer sitio— y cuyo aspecto (*su modo*) indica el efecto que tendrá la tecla pulsada. Como es posible teclear muchos símbolos distintos (incluyendo todas las palabras clave de BASIC) y el teclado sólo tiene 40 teclas, cada una de ellas significa algo diferente, dependiendo del modo del cursor y de cuál de las dos teclas de cambio de función —*shift*— se pulse. Algunos de los modos posibles se pueden provocar deliberadamente, mientras que otros son producidos automáticamente por el ordenador.

Las palabras clave *deben* teclearse utilizando los diversos modos y teclas de cambio de función, en lugar de deletrearse como en otras máquinas. Observe que los espacios, aunque admisibles, son innecesarios en general, porque el ordenador escribe las palabras clave insertando espacios en los lugares adecuados. Asimismo, en lo que respecta a los nombres de variables, se consideran equivalentes las letras mayúsculas y las minúsculas.

Modo L (letras). Esta es la condición que podríamos considerar normal, utilizada para letras minúsculas, letras mayúsculas (con CAPS SHIFT), dígitos, diversas teclas de control de edición (también con CAPS SHIFT), símbolos marcados con color rojo en las teclas (con SYMBOL SHIFT), ENTER y SPACE.

Modo C (mayúsculas permanentemente). Este modo es idéntico al anterior, exceptuando que las letras aparecen siempre en mayúsculas. Para pasar de modo L a modo C y viceversa, se pulsa CAPS LOCK (teclas '2' y 'CAPS SHIFT').

Modo K (palabras clave). Cuando el ordenador espera la introducción de una palabra clave de sentencia, o instrucción, entra en este modo sustituyendo automáticamente al modo L o al C: al principio de una línea (posiblemente después de su número), después de THEN o ':' (excepto en cadenas). Es igual que el modo L, con la diferencia de que las teclas alfabéticas pulsadas sin tecla de cambio (*shift*) producen palabras clave en lugar de letras. El modo K no se produce con datos para sentencias INPUT. Hay un mensaje informativo en forma de

cursor en modo K, que desaparece tan pronto como se pulse alguna tecla.

Modo E (extendido). Se obtiene pulsando simultáneamente las dos teclas SHIFT, y permite teclear las palabras clave escritas en verde y rojo encima y debajo de la tecla, respectivamente. Para la roja hace falta un cambio (con la mayoría de las teclas puede utilizarse indistintamente CAPS SHIFT o SYMBOL SHIFT, pero para las teclas de dígitos —fila superior— es obligatorio emplear SYMBOL SHIFT). El modo E sirve también para control de colores (apartado 3.3). Después de haber tecleado algo en modo E, o si se pulsan de nuevo las dos teclas SHIFT, el cursor vuelve a entrar en modo L, C o K.

Modo G (gráficos). Este modo se utiliza para producir los ocho caracteres gráficos de las teclas de dígitos, los caracteres inversos (pulsando también una tecla SHIFT), y con las teclas alfabéticas, los caracteres gráficos definidos por el usuario. Para cambiar a modo G desde los modos L, C y K, o viceversa, se teclaea GRAPHICS ('CAPS SHIFT' y '9'). De todas las funciones de edición, DELETE (borrado) sigue funcionando en modo G.

La edición de una línea se realiza —en modo L, C o K— con las teclas de movimiento del cursor ⇐ (5 cambiado) y ⇒ (8 cambiado) y la tecla DELETE (0 cambiado), cuyo efecto consiste en eliminar el carácter o palabra clave que haya a la izquierda del cursor. Tecleando se inserta un carácter a la izquierda del cursor, sin superponerse sobre lo que esté escrito.

Para borrar una línea entera se pulsa EDIT (1 cambiado), seguida, en caso necesario, por ENTER.

3.3. Caracteres de control de color

El procedimiento habitual de controlar los colores se describe en la Sección 7.

Se puede cambiar el color del programa utilizando los caracteres de control de color, tecleados directamente en el teclado en modo E. Su efecto no se observa inmediatamente, sino que cambian el color del texto que siga a continuación. Se puede controlar el color del «papel» (fondo) y de la «tinta» (escritura), el brillo, y pueden producirse también caracteres intermitentes.

En modo E, los dígitos del 1 al 7 (inclusive) y el 0 controlan el color del fondo o el de la escritura (con CAPS SHIFT); las teclas 8 y 9 sirven para especificar brillo normal o intenso, y acompañadas de CAPS SHIFT, determinan el carácter fijo o intermitente. Observe que, a diferencia de las teclas alfabéticas en modo E, las teclas numéricas poseen diferentes significados según cuál sea la tecla de cambio que se pulse (CAPS SHIFT o SYMBOL SHIFT).

El cursor se relaciona de forma muy peculiar con las teclas de control de colores, y es preciso emplear $\Rightarrow$ o DELETE dos veces con ellas. No se desanime por las cosas raras que ocurran en el proceso.

3.4. Instrucciones de manejo del teclado

Este apartado no pretende dar una descripción completa del teclado, sino proporcionar algunas indicaciones para ayudarle a orientarse. Conviene recordar una regla general bastante útil: pulse SYMBOL SHIFT para lo que esté escrito en rojo (aunque en Modo E puede utilizarse en su lugar CAPS SHIFT para los símbolos marcados debajo de las teclas alfabéticas).

Sentencias. Las sentencias más comunes se obtienen en modo K, en la misma fila que su letra inicial.

Hay dos grupos principales en modo E cambiado (debajo de las teclas, en rojo):

— Sentencias DEF FN (funciones definidas por el usuario) y sentencias de Interface 1 ZX, en la fila superior (debe emplearse SYMBOL SHIFT).

— BEEP (sonido), colores, etc., en la fila inferior.

Las demás sentencias están dispersas por todo el teclado:

NEW (nuevo), PAUSE (pausa), PLOT (trazar), DRAW (dibujar): Modo K, teclas A, M, Q y W respectivamente.

STOP (paro): Modo L, C o K, tecla A + SYMBOL SHIFT.

READ (leer), DATA (datos), RESTORE (restaurar),

LPRINT (imprimir), LLIST (listar en impresora): Modo E, teclas A, S, D, C y V respectivamente.

CIRCLE (círculo), VERIFY (verificar), MERGE (fundir), OUT (salida): Modo E, teclas H, R, T y O cambiadas (con SHIFT) respectivamente.

Operadores (incluyendo NOT). Todos ellos se obtienen en modo L, C o K con SYMBOL SHIFT (marcados en rojo en las teclas correspondientes).

Funciones. La mayoría se obtienen en el modo E con teclas sin cambiar (están en color verde sobre la tecla), y el resto en modo E cambiado (en color rojo, bajo la tecla):

- ASN (arco seno), ACS (arco coseno), ATN (arco tangente), VAL\$ (cadena equivalente), SCREEN\$ (imagen), ATTR (función de control de imagen), IN (lectura de entrada) — Q, W, E, J, K, L, I.
- POINT (punto), FN (función) — teclas 8 y 2 cambiadas (con SYMBOL SHIFT).

Instrucciones adicionales (por ejemplo, THEN, STEP). La mayoría se obtienen en modo L, C o K, con SYMBOL SHIFT (color rojo en las teclas).

Otros ejemplos son:

TAB (tabulación), BIN (binario): Modo E, P y B.

FN (función), LINE (línea): Modo E, SYMBOL SHIFT, 2 y 3.

3.5. Comandos

Todas las sentencias BASIC del Spectrum pueden utilizarse en líneas de comando, para ser ejecutadas de inmediato, o en líneas de programa, aunque algunas no tienen significado en comandos de ejecución inmediata (como por ejemplo, DATA o DEF FN).

Una línea puede contener varias sentencias, siempre que vayan separadas por dos puntos (:). No hay ninguna restricción impuesta al lugar que deben ocupar los diferentes tipos de sentencia en una línea, pero consulte al respecto las instrucciones IF y REM en el glosario.

Como es normal, no se ejecuta un comando hasta haber pulsado ENTER.

3.6. Programas

Si se empieza una línea con un número de línea (entero comprendido entre 1 y 9999), será insertada en el programa y no se ejecuta inmediatamente. El ordenador muestra automáticamente un listado del programa.

Los programas se ejecutan, normalmente, en respuesta al comando RUN (ejecución); sin embargo, RUN provoca el borrado de todas las variables y de la pantalla, para evitar lo cual en ocasiones es preferible ejecutar el programa por medio de una instrucción GO TO (ir a).

Para detener la ejecución de un programa en marcha se pulsa CAPS SHIFT y SPACE simultáneamente ('BREAK'). El ordenador investiga si se han pulsado estas teclas después de cada sentencia y durante las operaciones periféricas. Por otro lado, en determinadas circunstancias (por ejemplo, en el transcurso de operaciones con cassette), basta pulsar SPACE para interrumpir la ejecución del programa. Después de una interrupción deliberada ('BREAK'), o después de una detención debida a otras causas (como por ejemplo, un error), puede reanudarse el programa pulsando CONTINUE, que también funcionará aun cuando se hayan ejecutado otros comandos o se haya modificado el programa.

3.7. Edición de programas

Para borrar una línea de un programa, se teclea el número de la línea en cuestión y se introduce ('ENTER') sin más.

Para modificar una línea del programa, se teclea la nueva versión con el mismo número de línea (o se utiliza EDIT) y se pulsa ENTER a continuación.

EDIT (tecla 1 cambiada), hace aparecer una copia de una línea del programa en la parte inferior de la pantalla, de forma que puede ser modificada empleando el cursor.

La línea que reproduce el comando EDIT es la línea "en curso", que en general es la última línea insertada en el programa; esta línea está señalizada en los listados con el símbolo '>' situado inmediatamente después del número de línea. Las teclas $\downarrow$ y $\uparrow$ (6 y 7 cambiados) sirven para cambiar la línea en curso actual por la inmediata posterior o anterior respectivamente. Con el comando 'LIST' la línea en curso es la primera, y cuando se utiliza este mismo comando especificando un número de línea, es decir, 'LIST n.', la línea en curso pasa a ser la indicada.

En ocasiones se da el caso de que no existe línea en curso nominal (por ejemplo, después de borrar una línea o de ejecutar el comando LIST). No hay ninguna línea distinguida con la

marca '>', pero EDIT utiliza la línea de debajo, o al final del programa, la última línea.

3.8. Listados

Los listados se producen pulsando LIST (o LLIST), y también de forma automática después de modificar el programa, pulsar ↵ o ⬆, o introducir una línea en blanco.

Un listado producido en respuesta al comando LIST empieza por el principio o por la línea especificada, y prosigue hasta el final con desplazamiento de la imagen y la pregunta 'scroll?' (véase la Sección 8) en caso necesario.

El listado automático es más complicado. En efecto, su objetivo principal es visualizar la línea en curso (véase el apartado 3.7). Normalmente, el listado comienza por la misma línea que el último listado automático anterior y prosigue hasta llenar por completo la pantalla, incluyendo la línea en curso. Si para ello hace falta desplazar la imagen (*scrolling*), el listado empezará más abajo la próxima vez. Si el listado previo comenzó debajo de la que es ahora la línea en curso, el nuevo listado empezará por la línea en curso y llenará la pantalla.

4. Glosario de palabras BASIC

ABS argumento numérico

Función que da como resultado el valor absoluto del argumento.

ACS argumento numérico

Coseno inverso (arco coseno) del argumento, con el resultado expresado en radianes. Si el argumento no está dentro del margen comprendido entre -1 y $+1$, se obtendrá el mensaje de error 'A'.

AND

expresión numérica AND expresión numérica

expresión alfanumérica AND expresión numérica

Se trata de la operación lógica AND ("Y"). Su funcionamiento preciso difiere del de la operación lógica "Y"

convencional, y se explica con detalle en el apartado 5.7, pero combinando comparaciones en sentencias IF (si ...), tales como

IF edad > 150 AND x\$ = "todavía vive" THEN STOP

sí se comporta como es de esperar.

ASN argumento numérico

Seno inverso (arco seno) del argumento, con el resultado expresado en radianes. Si el argumento no está dentro del margen comprendido entre -1 y $+1$, se obtendrá el mensaje de error 'A'.

AT línea, columna

'línea' y 'columna' son expresiones enteras.

Se utiliza en sentencias PRINT, normalmente acompañada de un punto y coma (;) después de la expresión que indica la columna, a fin de que la salida siguiente comience a partir de la posición especificada. Por ejemplo:

PRINT AT 10,5;"excavadora"

Está diseñada para funcionar con cualquier dispositivo de salida (en PRINT#), y técnicamente, su efecto consiste en producir a la salida tres caracteres de control con los que se guía el dispositivo (véase el apartado 10.3).

Tanto el número de línea como el de columna se consideran en valor absoluto (es decir, se transforman en positivos si son negativos), y si alguno de ellos es mayor o igual que 256, se produce inmediatamente el error B.

Para el canal "s" (impresión —PRINT— normal en la parte superior de la pantalla de TV), las líneas están numeradas desde el 0, correspondiente a la línea superior, hasta el 23, correspondiente a la inferior (aunque las líneas 22 y 23 siempre están en la parte inferior de la pantalla), y las columnas lo están desde 0 —a la izquierda— hasta 31 —a la derecha—. Véase también el apartado 8.1.

Si el número de línea es mayor o igual que 23 o el de columna es mayor o igual que 32 se produce una situación de error B.

Si la línea está en la parte inferior de la pantalla (generalmente con el n.º 22, pero puede ser un número menor si se ha utilizado PRINT #1) se produce el error 5.

Para el canal "k" (PRINT #1 o INPUT —parte inferior de la pantalla—) la columna se numera como en la parte superior de la pantalla, pero el número de línea se mide en relación a la línea más alta de la parte inferior de la pantalla. Inicialmente (con sólo dos líneas), la línea 0 está en blanco y la línea 1 está justamente abajo, pero ambas se desplazan hacia arriba según se va ampliando la parte inferior de la pantalla con objeto de hacer sitio para más líneas. Véase también el apartado 8.2.

Si el número de línea es mayor o igual que 23, o el de columna es mayor o igual que 32, se produce una situación de error B.

Para el canal "p" (LPRINT —impresora ZX—) se ignora el número de línea, y el de columna modifica la posición de impresión en el sistema de almacenamiento intermedio (*buffer*) de la impresora. La instrucción AT nunca activa la impresora por sí misma. Véase también el apartado 8.4.

Si el número de columna es mayor o igual que 32 se produce el error B.

No tiene ningún efecto especial en los demás canales.

ATN argumento numérico

Tangente inversa (arco tangente) del argumento, con el resultado expresado en radianes.

ATTR (línea, columna)

'línea' y 'columna' son expresiones enteras que especifican la posición de un carácter en la pantalla de TV (de forma análoga a AT), y el resultado es un número comprendido entre 0 y 255 el cual, escrito en sistema binario, con 8 bits, codifica las características —atributos— de dicha posición:

x x x x x x x x

color de papel

(de 0 a 7)

color de tinta

(de 0 a 7)

0 = normal, 1 = brillante

0 = fijo, 1 = intermitente

por ejemplo, el número 56 (igual a 0 0 111 000 en sistema binario) describe un carácter negro (tinta negra) sobre fondo (papel) blanco, fijo (no intermitente) y de brillo normal. El número 178 (1 0 110 010 en binario) correspondería a un carácter escrito con tinta roja sobre papel amarillo, intermitente y de brillo normal. Si la línea o la columna tienen un valor absoluto igual o superior a 256 se produce el error B, y si están fuera de los márgenes normales de 0-23 y 0-31 respectivamente, el resultado de la función ATTR carece de significado.

BEEP duración, tono

'duración' y 'tono' son dos expresiones numéricas, que expresan la duración en segundos y el tono en semitono por encima o por debajo del DO central (los más agudos se representan por números positivos y los más graves por negativos).

BEEP 1.5,12 produce un DO mayor durante 1½ segundos.

Cuando no es válida la combinación de duración y tono se produce el error B (el mensaje 'Integer out of range' —número entero fuera del margen válido— puede originar confusiones, puesto que ni el valor de la duración ni el del tono han de ser números enteros). El número del tono debe ser mayor que -60 y menor que +70, mientras que la longitud tiene que estar comprendida entre 0 y 10 segundos, si bien para tonos agudos, por encima de 55, la máxima duración admisible empieza a disminuir gradualmente, hasta que en el tono 69 debe ser inferior a unos 4.7 segundos.

BIN número binario

Sirve para escribir un número con notación binaria; por ejemplo, 5 se escribirá 'BIN 101'.

Si el número es superior a 65535 (dieciséis 'unos' binarios) se produce un error de sintaxis o error C.

Los únicos símbolos válidos para escribir el número son el 0 y el 1. Si se escribe BIN sin ningún dígito a continuación será considerado como 0, lo cual puede ser engañoso en ejemplos tales como BIN + 11, el cual sería interpretado como 0 + 11, es decir, el decimal 11.

BORDER color

'color' es una expresión entera que se mantiene en el margen de 0 a 7 (de lo contrario se produciría el error K). Sirve para determinar el color del contorno, y también el color de fondo para la parte inferior de la pantalla.

BRIGHT código de brillo

el 'código de brillo' es una expresión entera que sólo puede adoptar los valores 0, 1 y 8 (cualquier otro valor da lugar al error K). Determina el valor de BRILLO de una combinación de colores, y se utiliza, bien como sentencia (para la combinación permanente de colores), bien como instrucción PRINT (para una combinación temporal). Véase el apartado 7.2.

CAT unidad

CAT# corriente; unidad

'unidad' y 'corriente' (*drive, stream*) son expresiones enteras. Estas sentencias sirven para catalogar los ficheros contenidos en un cartucho de Microdrive (véase el apartado 11.1).

La forma 'CAT' será aceptada, pero provoca un error 0. Esto puede ser un procedimiento conveniente para inicializar las variables virtuales del sistema en el Interface 1 ZX, aunque no suele ser necesario.

CHR\$ código

'código' es un argumento entero —un código ASCII— y el resultado de la función es una cadena compuesta por un solo carácter, el correspondiente al código especificado.

CIRCLE x, y, radio

CIRCLE elementos de color; x, y, radio

'x', 'y' y 'radio' son expresiones enteras que indican respectivamente las coordenadas x e y del punto de imagen (pixel) donde está centrado el círculo, y su radio, expresadas en las mismas unidades de medida. Los elementos de color se explicarán en el apartado 7.3. Con esta sentencia se dibuja una circunferencia de alta resolución; por ejemplo, CIRCLE 120,100,50 dibuja un círculo de radio 50 con el centro en (120,100); CIRCLE INK 4;120,100,50 dibuja el mismo círculo pero en verde.

Después de dibujar el círculo, la posición de trazado (posición PLOT) (para una sentencia DRAW posterior) queda establecida aproximadamente en la mitad de la parte inferior derecha del círculo. Si se intenta dibujar un círculo que se salga de la parte superior de la pantalla se produce el error B.

CLEAR

CLEAR ramtop

'ramtop' es una expresión entera utilizada para definir la dirección de memoria más alta del sistema BASIC. Esta sentencia sirve para recuperar el espacio de la memoria ocupado por variables borrándolas, de forma que se produce el error 2 si se incluyen en una expresión. También borra la pantalla (CLS) y la pila GOSUB (por si acaso había más GOSUBs que RETURNS).

Si se especifica 'ramtop', la parte de la memoria definida por direcciones superiores a ella queda disponible para cualquier aplicación que desee el usuario, tal como definición de gráficos, código de máquina, etc.

Si se especifica como límite de RAM ('ramtop') una dirección que exceda de la memoria existente (es decir, 32768 o más en el modelo de 16K), o que sea insuficiente para contener al menos 50 bytes libres, se produce el error M. Cuando se produce este error, tanto las variables como la pantalla habrán sido ya borradas, no así la pila GOSUB.

'CLEAR 0' no da lugar a error, sino que es equivalente a 'CLEAR'.

Contrariamente al manual, CLEAR no ejecuta una restauración (RESTORE).

CLEAR

Esta sentencia sólo es utilizable con el Interface 1 ZX conectado. Cierra todas las corrientes (*streams*) y luego asigna a las corrientes estándar nos. 0, 1, 2 y 3 sus canales convencionales. Es posible que no cierre correctamente los ficheros de 'escritura' y por lo tanto, no debe utilizarse con ellos.

CLOSE # corriente

'corriente' es una expresión entera. Esta sentencia cierra el canal ligado a la corriente que se especifica, dejándolo sin

ningún canal, exceptuando las corrientes estándar —nos. 0, 1, 2 y 3— a las cuales se vuelven a conectar los respectivos canales convencionales. Véase la Sección 10.

La sentencia **CLOSE #** sólo se aplica completamente con el Interface 1 ZX; sin él no tiene sentido cerrar una corriente que no ha sido abierta.

CLS

Borra la pantalla dejando una imagen que sólo contiene espacios, con los atributos de la combinación permanente de colores (excepto la parte inferior de la pantalla, que utiliza los suyos propios, basados en el color del contorno). Así pues, con **PAPER 6: INK 1: FLASH 1: CLS** el color de la parte superior de la pantalla cambiaría entre amarillo y azul.

Para los valores 8 y 9 de combinación de colores, CLS puede dar resultados inesperados, dependiendo de las anteriores combinaciones de color.

CLS

Esta sentencia sólo tiene aplicación con el Interface 1 ZX.

Es similar a CLS, pero en primer lugar ejecuta **INK 0:**

PAPER 7: FLASH 0: BRIGHT 0: OVER 0: INVERSE 0, de modo que la combinación de colores es negro sobre blanco, la misma que hay cuando se enciende el aparato.

CODE argumento alfanumérico

Esta función da como resultado el código ASCII del primer carácter de la cadena, y 0 si se trata de una cadena nula (vacía).

La palabra **CODE** (código) se utiliza también en **SAVE** (grabar), **VERIFY** (verificar), **LOAD** (cargar) y **MERGE** (fundir) para archivos de bytes.

CONTINUE

(**CONT** en el teclado). Esta sentencia (**CONT**inuar) sirve para reanudar la ejecución de un programa después de que haya sido detenida. Los números de línea y sentencia que aparecen en el último mensaje informativo indican el punto donde se reanudará el programa, a excepción de los mensajes tipo '0 OK', que no se tienen en cuenta. Esto permite al usuario ejecutar comandos —para examinar una variable,

por ejemplo— antes de pulsar CONT para seguir adelante con el programa.

Generalmente se repite la sentencia donde se detuvo el programa, porque se da por supuesto que la interrupción ha sido provocada por un error, pero si el mensaje es 9 (STOP) o L (interrupción entre sentencias producida pulsando BREAK), el programa continuará con la sentencia siguiente, después de realizar los saltos necesarios.

Es perfectamente válido modificar el programa antes de utilizar CONT, aunque es raro que funcione correctamente si el error ha tenido lugar en un comando, ya que habrá desaparecido desde entonces.

COPY

Envía una copia de las 22 líneas superiores de la pantalla de TV a la impresora ZX, si está conectada (si no lo está no produce ningún efecto).

COS ángulo

‘ángulo’ es un argumento numérico expresado en radianes (no en grados). La función calcula el coseno del ángulo.

DATA lista de datos

La lista de datos es una serie de expresiones separadas por comas, como por ejemplo: 50 DATA 10.3, “conejo”, 2 * a(4), CHR\$ 0 + “robot”.

Las sentencias DATA incluidas en el programa (DATA no se emplea en líneas de comando) proporcionan los datos leídos por READ (leer). A diferencia de otras versiones de BASIC, las cadenas incluidas en la lista de datos deben ir entrecomilladas.

La palabra DATA se utiliza también en SAVE, VERIFY, LOAD y MERGE para archivos de matrices.

DEF FNF nombre de función (lista de nombres de argumentos)
= expresión

Ejemplo: 10 DEF FN s(x) = x * x

Sentencia que se utiliza para definir una función definida por el usuario (tal como FN s en el ejemplo anterior). Véase el apartado 5.8.

DIM nombre de matriz (lista de dimensiones)

El nombre de la matriz está compuesto por una letra si se trata de una matriz numérica, y de una letra seguida por el signo \$ si la matriz es alfanumérica (compuesta por caracteres). La lista de dimensiones es una serie de expresiones enteras separadas por comas (véase el apartado 5.4). Ejemplo: DIM a(20):DIM b\$(15,6,9). Si las dimensiones no están comprendidas entre 1 y 65535 se produce el error B.

DRAW x,y

DRAW elementos de color; x,y

DRAW x,y, ángulo

DRAW elementos de color; x,y, ángulo

'x' e 'y' son expresiones enteras que indican, respectivamente, el número de pixels hacia la derecha y hacia arriba (contados a partir de la posición inicial) donde se encuentra el punto en el que termina el dibujo (**DRAW**); 'ángulo' es una expresión numérica que expresa en radianes el ángulo de giro (en sentido contrario al movimiento de las agujas del reloj). Los 'elementos de color' serán explicados en el apartado 7.3. Por ejemplo, **DRAW** 53,—60: **DRAW** INK 5; 40,30,—PI/2.

Esta sentencia dibuja un arco de circunferencia o un segmento de línea recta. Empieza donde terminó la última sentencia **PLOT**, **DRAW** o **CIRCLE**, termina en el pixel definido por las coordenadas x e y (relativas al pixel inicial), y efectúa un giro igual al ángulo especificado. Cuando no se especifica el ángulo, se considera éste nulo, de modo que **DRAW** traza una línea recta. Véase también el apartado 7.3.

Se incurre en un error B si x o y no están dentro del margen delimitado por -255 y +255, y cuando la figura se sale de la parte superior de la pantalla mientras se están dibujando.

ERASE "m"; n.º de unidad; nombre de archivo

Esta sentencia sólo se utiliza con el Microdrive, y sirve para borrar un archivo de uno de los cartuchos (véase el apartado 11.1).

Si la sentencia tiene la forma 'ERASE expresión alfanumérica' será aceptada, pero producirá un error 0.

EXP argumento numérico

La función exponencial EXP es igual a e^x , donde e es la denominación matemática de una constante que vale aproximadamente 2.718281828.

Si el argumento es superior a 88 se produce el error 6.

FLASH código de intermitencia

'código de intermitencia' es una expresión entera que puede adoptar únicamente tres valores: 0, 1 u 8 (de lo contrario se incurre en el error K). Sirve para definir la característica de intermitencia de una combinación de colores (ver 7.2), y puede utilizarse como sentencia (para la combinación permanente) o como elemento PRINT (para combinaciones temporales).

FN nombre de función (lista de argumentos)

Sirve para utilizar una función previamente definida por el usuario en una sentencia DEF FN; por ejemplo: PRINT 3 + FN s(9/7).

Véase el apartado 5.8.

FOR variable = valor inicial TO valor final

FOR variable = valor inicial TO valor final STEP paso

Ejemplos:

```
50 FOR n = 1 TO 10
```

```
100 FOR m = 0.1 TO 0.9 STEP 0.01
```

'valor inicial', 'valor final' y 'paso' son expresiones numéricas; la variable debe ser una variable numérica simple cuyo nombre conste de una sola letra. Véase el apartado 6.3.

FORMAT "m"; unidad; nombre de cartucho

Sentencia para determinar el formato de un cartucho de Microdrive (11.1).

FORMAT "n"; número de estación

Sentencia para determinar el número de estación en la red (11.2).

FORMAT "b"; baudios

FORMAT "t"; baudios

Sentencia para determinar la velocidad de transmisión (medida en baudios) del interface RS232 —véase el apartado 11.3.

Una sentencia de forma 'FORMAT' expresión alfanumérica' será aceptada, pero dará lugar a un error 0.

GO SUB número de línea

El 'número de línea' puede ser cualquier expresión entera, igual que sucede con GO TO. Consulte el apartado 6.4.

GO TO número de línea

'número de línea' puede ser cualquier expresión entera. Si el número de línea no corresponde a ninguna línea del programa, la bifurcación se produce pasando a la línea que tenga el número inmediatamente superior al indicado. Por ejemplo, en

```
10 GO TO 30
20 PRINT "pasivo"
40 PRINT "conejo"
```

GO TO provoca un salto a la línea 40. Si GO TO tiene un número que sobrepasa el final del programa, normalmente se detendrá la ejecución con el mensaje 'OK', pero si está conectado el Interface 1 ZX es posible que aparezca el mensaje 'Program finished' (programa terminado).

IF condición THEN sentencias

Por ejemplo, IF INKEY\$ < > "" THEN PRINT "?"; GO TO 5.

Véase el apartado 6.2.

IN dirección de port

La 'dirección de port' es un argumento entero comprendido entre 0 y 65535 (en caso contrario se obtendría error B). Esta función lee el port de entrada al procesador Z80A cuya dirección se especifica, siendo el resultado el byte de entrada, entre 0 y 255. La función IN puede utilizarse con los ports internos del Spectrum (por ejemplo, para leer directamente el teclado) y con cualquier port empleado por dispositivos periféricos. Sin embargo, no tiene nada que ver con los canales y corrientes utilizados por el Interface 1 ZX, los cuales se estudian en la Sección 9.

Existen realmente 65536 ports, y no 256, porque todos los 16 bits se colocan en el bus de direcciones del procesador Z80A.

INK código de color

'código de color' es una expresión entera de valor comprendido entre 0 y 9 (otros valores darían lugar a un error K). Sirve para determinar el valor INK (color de escritura) de una combinación de colores. Se utiliza tanto en forma de sentencia (para la combinación permanente) como de comando PRINT (para una combinación temporal). Véase el apartado 7.2.

INKEY\$

Se trata de una función sin argumento utilizada para leer el teclado. El resultado es una cadena, que puede estar compuesta por un solo carácter (indicando la tecla que se ha pulsado) o por ninguno —cadena vacía— si el teclado no muestra ningún carácter o más de uno. Considera adecuadamente las dos teclas de cambio (shift), así como el Modo C, en caso de que esté activado. Las teclas de la fila superior dan los siguientes resultados cuando están cambiadas:

EDIT (1)	CHR\$ 7
CAPS LOCK (2)	CHR\$ 6
TRUE VIDEO (3)	CHR\$ 4
INVERSE VIDEO (4)	CHR\$ 5
⇐ (5)	CHR\$ 8
↓ (6)	CHR\$ 10
↑ (7)	CHR\$ 11
⇒ (8)	CHR\$ 9
GRAPHICS (9)	CHR\$ 15
DELETE (0)	CHR\$ 12
CAPS SHIFT y SYMBOL SHIFT juntas	CHR\$ 14

INKEY\$ # número de corriente

'número de corriente' es un argumento entero. INKEY\$ y el símbolo # se teclean por separado. Esta función toma un carácter de la corriente especificada, y da como resultado precisamente dicho carácter. Si no hay ningún carácter disponible de inmediato, caben dos posibilidades:

1. No habrá ninguno disponible, en cuyo caso INKEY\$.# da error 8. Esta situación puede producirse con un canal del Microdrive, cuando el archivo ha sido leído en su totalidad, y también en la red, cuando Iris ha cerrado su canal.

2. Habrá caracteres disponibles, pero no han llegado todavía. Entonces, el resultado de `INKEY$ #` es una cadena nula (igual que en `INKEY$` cuando no se pulsa ninguna tecla). Esto puede ocurrir en un canal del RS232, o en un canal de la red, cuando Iris ha dejado abierto su canal pero no está transmitiendo en el momento.

Con un canal de red de transmisión, `INKEY$ #` esperará indefinidamente la llegada de su carácter.

La relación recíproca entre el teclado e `INKEY$ #` (`#0` ó `#1`) no es totalmente correcta; la función tiene su aplicación en los canales del Interface 1 ZX. Consulte el apartado 10.4 y la Sección 11.

Errores: B si la corriente está fuera del margen 0-15, y 0 si la corriente no está abierta.

`INPUT` etc.

Por ejemplo, `INPUT n`

El ordenador Spectrum es muy flexible en lo que respecta a la impresión de mensajes y advertencias en la parte inferior de la pantalla; por ejemplo:

```
100 INPUT "Como te llamas?";n$;"Que edad tienes?";edad
```

Ver apartado 8.3. `INPUT` se utiliza también con `#` para obtener datos de dispositivos periféricos. Por ejemplo:
`INPUT # 6; teléfono`

Ver apartado 10.4 y la descripción de `LINE` (utilizada como en `INPUT LINE a$`).

Errores. Todos los errores que pueden cometerse con `LET` o `PRINT` son también posibles con `INPUT`.

H — Existe la posibilidad de provocar una interrupción en medio de una sentencia `INPUT` introduciendo `STOP` como primer carácter tecleado, con lo cual el programa se detiene mostrando el mensaje H. Pulsando `CONT` se repite la sentencia `INPUT`.

`INT` argumento numérico

Función cuyo resultado es la parte entera de su argumento.

No se obtiene el número entero más aproximado, sino el *menor* número entero más aproximado.

Ejemplos: `INT 5.99 = 5`, `INT 5.01 = 5`, `INT 5 = 5`, `INT -4.99 = -5`.

INVERSE código de inversión

'código de inversión' es una expresión entera con dos únicos valores posibles: 0 y 1 (cualquier otro produce un error K). Sirve para establecer el valor INVERSE (inverso) de una combinación de colores (véase el apartado 7.2), y se utiliza como sentencia (para la combinación permanente) o como elemento PRINT (para la combinación temporal).

LEN argumento alfanumérico

Función cuyo resultado es el número de caracteres de que consta una cadena. Por ejemplo:

LEN "seis" = 4, LEN "" = 0

LET variable = expresión

Por ejemplo: LET precio = 129.95:LET a\$(3,2 TO 5) = "Pala".

A diferencia de otras versiones de BASIC, no puede omitirse la palabra clave LET en el Spectrum.

LINE variable alfanumérica

Puede emplearse en sentencias INPUT, con una función análoga a la de LINPUT en otras versiones de BASIC. Por ejemplo: INPUT a\$, LINE b\$.

a\$ y b\$ son introducidas de forma distinta. Para a\$, el ordenador espera una expresión alfanumérica (cadena) correcta, y hace aparecer automáticamente las comillas en la pantalla, mientras que para b\$ no pone comillas, sino que toma todos los caracteres que haya tecleado el usuario y los utiliza directamente en la cadena; las comillas que acompañan a toda cadena las pone al final, encerrando todo lo que se ha tecleado y duplicando las que hubiera en la línea de entrada. A continuación presentamos algunos ejemplos con sentencias PRINT que ilustran los distintos métodos.

INPUT		PRINT	
a\$	LINE b\$	a\$	b\$
"callado"	callado	callado	callado
"cal" + "lado"	"cal" + "lado"	callado	"cal" + "lado"
""cal""	"cal"	"cal"	"cal"

LIST

LIST número de línea

'número de línea' es una expresión entera (si no se especifica se considera igual a 0, lo que significa la primera línea). Además de producir un listado del programa a partir de la línea indicada, transforma dicha línea en la línea en curso (ver apartados 3.6 y 3.8).

LIST # corriente

LIST # corriente, número de línea

Por ejemplo: LIST # 6,100.

Es similar a LIST, pero envía el listado a la corriente especificada.

LLIST

LLIST número de línea

Es similar a LIST, pero utiliza la corriente n.º 3 (impresora ZX). También es válida la forma LLIST # ...

LN argumento numérico

Función que calcula el logaritmo natural (neperiano, o de base e). Por ejemplo:

LN EXP 5.3 = 5.3.

Si el argumento es 0 o negativo se incurre en error A.

LOAD etc

Por ejemplo: LOAD "imagen" SCREEN\$: LOAD "prog".

Esta sentencia sirve para cargar el ordenador desde una cinta cassette u otro periférico (10.5).

LPRINT etc

Es similar a PRINT, pero utiliza la corriente n.º 3 (impresora ZX). Véase el apartado 8.4.

MERGE etc

Sentencia para fundir (combinar) un programa nuevo con uno anterior, ya sea a partir de una cinta cassette (véase la Sección 9) o desde otro periférico (10.5).

MOVE origen TO destino

Sólo tiene aplicación con el Interface 1 ZX. 'origen' y 'destino' representan corrientes (utilizando el símbolo #) o canales (con el indicativo apropiado de canal). Por ejemplo, MOVE "M";2;"compras" TO #2 escribirá en la pantalla de TV su archivo de compras almacenado en Microdrive. La sentencia copia caracteres del 'origen' en el 'destino'

continuando hasta agotar el primero. Si se emplean corrientes, deben estar previamente abiertas, pero los canales que se especifican directamente son abiertos automáticamente al principio de MOVE y se vuelven a cerrar después. Consulte también el apartado 10.6 y la Sección 11.

No deben emplearse las letras "k", "s" o "p" para especificar los canales convencionales; se utilizan corrientes en su lugar. Tenga cuidado al utilizar #0 ó #1 como origen, porque MOVE no terminará espontáneamente y tal vez se vea incapaz para provocar la interrupción (BREAK).

Una sentencia de formato 'MOVE cadena, cadena' será aceptada, pero dará lugar a un error 0.

NEW

Su efecto consiste en borrar todo el sistema BASIC —programa, variables, etc.— pero hasta la dirección RAMTOP de la memoria (véase CLEAR), es decir, se conservan los gráficos definidos por el usuario y los datos almacenados en direcciones superiores a RAMTOP.

NEXT variable

Véase el apartado 6.3. El error 1 significa que, a pesar de haber encontrado la variable, ésta no ha sido establecida en una sentencia FOR.

NOT expresión numérica

Función lógica (véase el apartado 5.7) que invierte el resultado de las condiciones. Verbigracia: 'IF NOT a = 6 ...' significa lo mismo que 'IF a < > 6 ...'; 'IF NOT (x * x < 2 AND p\$ = "y") ...' significa lo mismo que 'IF x * x > = 2 OR p\$ < > "y" ...'

OPEN # corriente, canal

OPEN # corriente; canal

'corriente' es una expresión entera de valor comprendido entre 0 y 15 (aunque Sinclair aconseja no utilizar los valores 0, 1, 2 y 3), y 'canal' es un indicador de canal apropiado al dispositivo con el que se trabaja. Por ejemplo:

OPEN #5,"s"

(TV, parte superior de la pantalla)

OPEN #4;"m";3;"cuentas" (un archivo almacenado en microdrive)

Esta sentencia abre el canal y lo conecta a la corriente (ver Secciones 10 y 11).

Si el número de corriente es igual o mayor que 16 se obtiene el error 0, o bien el mensaje 'Invalid stream number' (número de corriente inválido); el mensaje 'Stream already open' (corriente ya abierta) aparece cuando la corriente en cuestión ya tenía asignado un canal (que no fuese uno de los convencionales "k", "s" o "p"). En tal caso debe cerrarse antes de poderlo volver a abrir.

Sin el Interface 1 ZX es obligatorio utilizar el formato con coma, mientras que con dicho interface se puede emplear coma (,) o punto y coma (;), teniendo presente que un punto y coma con "k", "s" o "p" puede causar un error identificado por medio del mensaje 'Invalid device expression' (expresión inválida de dispositivo).

OR

expresión numérica OR expresión numérica

Operación lógica OR (O). Su comportamiento preciso es algo distinto del convencional (véase el apartado 5.7), pero cuando se utiliza con comparaciones incluidas en sentencias IF tales como 'IF reserva < 5 OR precio < = 17 ...' produce los resultados que cabría esperar.

OUT port, valor

'port' y 'valor' son expresiones enteras, limitada la primera entre 0 y 65535 y la segunda entre -255 y +255 (si alguna se sale del margen respectivo dará lugar a error B).

Envía un valor a un port de salida del procesador Z80A (ver IN).

OVER código de sobreimpresión

El 'código de sobreimpresión' es una expresión entera que puede adoptar únicamente los valores 0 y 1 (con otro valor se incurre en error K). Sirve para determinar si los caracteres nuevos borran los antiguos o si se combinan ambos (sobreimpresión); véase el apartado 7.2. Puede utilizarse como sentencia —para la combinación permanente de colores— o como elemento PRINT —para una combinación temporal.

PAPER código de color

'código de color' es una expresión entera de valor comprendido entre 0 y 9 (so pena de incurrir en error K). Establece el valor del color de fondo (papel) en una combinación de colores (7.2), utilizada como sentencia, para una combinación permanente, o como elemento PRINT, para una temporal.

PAUSE tiempo

'tiempo' es una expresión entera, tal que cada unidad representa 1/50 avo de segundo. El ordenador deja transcurrir el tiempo indicado antes de proseguir. PAUSE 0 no se cuenta, sino que provoca una pausa indefinida. En cualquier caso, si se pulsa una tecla que no sea de cambio —shift— durante la pausa, el programa se reanuda inmediatamente. Es decir, PAUSE 50 produce una pausa de un segundo, o menos, si se pulsa una tecla.

Si su modelo de Spectrum ha sido diseñado para un país en el que la televisión funcione a razón de 60 imágenes por segundo en lugar de 50, debe especificar el tiempo en 1/60 avos de segundo, aproximadamente.

PEEK dirección de la memoria

'dirección de la memoria' es un argumento entero comprendido entre 0 y 65535 (si se sale de este margen se produce un error B). La finalidad de esta función es leer un byte de la memoria.

PI

Número π (3.14159...).

PLOT x, y

PLOT elementos de color; x, y

Ejemplo:

PLOT 128,88:PLOT INK 5;9,10

'x' e 'y' son expresiones enteras que definen las coordenadas de un pixel (tomando sus valores absolutos, o dicho en otras palabras, transformándolas en números positivos). x está limitado entre 0 y 255, mientras que por su parte, y lo está entre 0 y 175 (en caso contrario se daría lugar al error B). Inscribe en la pantalla un pixel de alta resolución,

transformándolo normalmente en color de tinta. Ver apartado 7.3.

POINT (x, y)

'x' e 'y' son expresiones enteras que definen las coordenadas de un pixel; se encuentran comprendidas entre 0 y 255, y entre 0 y 175, respectivamente (en otro caso se produce un error B). El objetivo de esta función es investigar si el pixel especificado está o no ocupado: da como resultado 1 si es tinta, y 0 si es papel. Puede emplearse con el formato 'IF POINT (x,y) THEN...' de forma que se ejecuta la instrucción introducida por THEN si el punto está dibujado (es decir, si es tinta).

POKE dirección de la memoria, valor

'dirección de la memoria' y 'valor' son dos expresiones enteras, estando la dirección comprendida entre 0 y 65535 y el valor entre -255 y +255 (si no lo están se produce un error B). Esta sentencia carga un byte en la dirección especificada.

PRINT etc

Véase el apartado 8.1. PRINT sirve también para enviar datos a dispositivos periféricos, utilizando el símbolo # (ver 10.4).

RANDOMIZE

RANDOMIZE 'número iniciador'

(RAND en el teclado)

El 'número iniciador' es una expresión entera. Estas sentencias controlan la función RND haciéndola que produzca secuencias fijas de números aparentemente aleatorios, lo cual resulta especialmente útil para corregir (depurar) programas; en efecto, RANDOMIZE 5 hace que RND siga una secuencia fija y RANDOMIZE 243 da lugar a otra secuencia fija distinta. RANDOMIZE 0 y RANDOMIZE son diferentes: RND comienza en un valor realmente aleatorio que depende del tiempo que lleve conectado el ordenador.

READ variables separadas por comas

Ejemplo:

READ n,a(n,2),p\$(5 TO 8)

Asigna valores tomados de las sentencias DATA (las expresiones DATA son evaluadas cuando las necesita READ).

Errores: C si en las expresiones no se corresponden números con números y cadenas con cadenas; E si no hay suficientes sentencias DATA —o si se ha olvidado poner un RESTORE.

REM etc

Recordatorio o nota. Los dos puntos (:) se consideran como parte de la sentencia REM, y por consiguiente sólo puede ser la última sentencia de una línea. Por ejemplo:

170 REM todo esto se ignora, incluyendo esta aparente sentencia PRINT:PRINT “no se imprime”.

RESTORE

RESTORE número de línea

‘número de línea’ es una expresión entera. Su efecto consiste en hacer que las sentencias READ posteriores empiecen a leer por la línea especificada (o por el principio del programa, si no se especifica línea alguna).

No es válido especificar números de línea mayores que 9999, pues es posible que el programa no funcione.

RETURN

Véase 6.4. Se produce el error 7 si el ordenador no está ejecutando una subrutina introducida por GOSUB al llegar a RETURN.

RND

Esta función sirve para producir un número pseudoaleatorio (sólo *parece* aleatorio) comprendido entre 0 y 1, sin llegar nunca a valor 1.

Por ejemplo:

LET d = INT (RND *6) + 1

produce un número “aleatorio” entre 1 y 6. A diferencia de otras versiones de BASIC, RND carece de argumento.

RUN

RUN número de línea

'número de línea' es una expresión entera. Este comando ejecuta un borrado (CLEAR), una restauración (RESTORE) y después salta a la línea cuyo número se indica, o a la primera del programa cuando no se indica ninguno.

Si está conectado el Interface 1 ZX y se pulsa RUN inmediatamente después de la puesta en marcha o de un comando NEW, el ordenador Spectrum buscará en la unidad 1 del Microdrive un programa llamado "run", lo cargará, y si fue grabado mediante SAVE... LINE, lo ejecutará. En caso contrario, con el Interface 1 ZX conectado pero sin programa, RUN puede dar como resultado la aparición del mensaje 'Program finished'.

SAVE etc

Instrucción para efectuar grabaciones en cassette (Sección 9) y algunos otros dispositivos periféricos (10.5).

SCREEN\$ (línea, columna)

'línea' y 'columna' son expresiones enteras que definen una posición en la pantalla; 'línea' puede valer de 0 a 23 y 'columna' de 0 a 31 (en otro caso se produce un error B). Esta función sirve para averiguar el carácter que ocupa la posición especificada. En caso de caracteres no reconocibles (caracteres de mosaico y gráficos definidos por el usuario —códigos 128 a 164— y figuras obtenidas por medio de sobreimpresión o trazado) SCREEN\$ da como resultado la cadena vacía "". Los caracteres en vídeo inverso son reconocidos como sus versiones normales.

La palabra clave SCREEN\$ se utiliza también en instrucciones SAVE y LOAD para un archivo de bytes que registra el archivo de presentación en pantalla.

SGN argumento numérico

Función para calcular el signo de un número; su resultado es +1, -1 ó 0, según que el número sea positivo, negativo o cero. Ejemplos:

$\text{SGN } 7.2 = 1$, $\text{SGN } 0 = 0$, $\text{SGN } -9 = -1$.

SIN ángulo

'ángulo' es un argumento numérico expresado en radianes (no en grados). Esta función calcula el seno del ángulo.

SQR argumento numérico

Función para calcular la raíz cuadrada; por ejemplo, $\text{SQR } 9 = 3$. Si el argumento es negativo se incurre en el error A.

STEP

(Paso) Se utiliza en sentencias FOR. Véase el apartado 6.3.

STOP

Detiene el programa, apareciendo en la pantalla el mensaje informativo 9. Para reanudar la ejecución interrumpida se pulsa CONT. STOP es utilizada también en introducción de datos con INPUT para provocar interrupciones acompañadas del mensaje H, y sirve como respuesta al mensaje 'scroll?' (¿desplazar pantalla?) con el significado 'no desplazar'.

STR\$ argumento numérico

Función para producir una cadena compuesta por los caracteres que aparecerían si se imprimiese el argumento numérico. Por ejemplo, $\text{STR\$ } 2.1e + 2 = "210"$.

TAB columna

'columna' es una expresión entera. Esta palabra se utiliza en sentencias PRINT, seguida normalmente por un punto y coma (;) con objeto de que la siguiente salida empiece a imprimirse en la columna especificada. Por ejemplo:

PRINT a;TAB 8;b;TAB 16;c

Está concebida para funcionar con cualquier dispositivo de salida, y técnicamente, su efecto consiste en producir a la salida tres caracteres de control para guiar al dispositivo (véase el apartado 10.3).

Para los canales "s" (PRINT normal, en la parte superior de la pantalla), "k" (PRINT #1 o INPUT —parte inferior de la pantalla—) y "p" (LPRINT —impresora ZX—), TAB produce espacios en número suficiente como para que la impresión comience en el lugar especificado.

Si el número de columna es 32 o superior, se divide entonces por 32 y se toma el resto obtenido. Otros dispositivos no reaccionan ante TAB (ni siquiera el canal "t", donde podría ser útil).

TAN ángulo

'ángulo' es un argumento numérico expresado en radianes (no en grados).

Esta función calcula la tangente trigonométrica del ángulo. Se produce el error 6 si el valor del ángulo se acerca demasiado a $\pi/2$, ángulo cuya tangente es infinita.

THEN

Forma parte de sentencias IF. Véase el apartado 6.2.

TO

Se utiliza en sentencias FOR y MOVE, así como en fragmentación de cadenas (véase el apartado 5.5).

USR dirección de la memoria

'dirección de la memoria' es un argumento entero. La función ejecuta una subrutina escrita en lenguaje de máquina, cuya dirección inicial es la indicada. Produce un resultado que es el valor del par de registros BC del procesador Z80A (entre 0 y 65535) después de ejecutada la subrutina —al 'volver' al programa (*return*). El lugar más idóneo para conservar subrutinas escritas en código de máquina es la zona próxima al límite superior de la memoria, recurriendo a CLEAR para reservar espacio.

USR argumento alfanumérico

Aplicación no convencional de USR pensada para ayudar a la creación de caracteres gráficos definidos por el usuario. La cadena especifica un gráfico definido por el usuario: se trata de un solo carácter que puede ser, bien el propio gráfico, bien la letra correspondiente, de la A a la U (mayúscula o minúscula). Si no es ninguna de estas cosas se incurre en el error A.

El resultado de la función es la dirección de la memoria que contiene el primero de los 8 bytes que definen la forma del gráfico; verbigracia (en un modelo Spectrum de 48K recién encendido):

USR "a" = 65368, USR "b" = 65376, etc.

Si se cambian de sitio las definiciones de los caracteres y se modifica la variable UDG (User Defined Graphics —gráficos definidos por el usuario) del sistema para tenerlo en cuenta,

USR dará los valores nuevos, ya corregidos. Véase el apartado 7.4.

VAL argumento alfanumérico

Esta función toma los caracteres que forman una cadena y los trata como si fuesen una expresión numérica; en efecto, el resultado es el valor numérico de la expresión. Por ejemplo:

VAL "2.3" = 2.3, VAL "2 + 4" = 6, VAL
("2" + "4") = VAL "24" = 24.

Si la cadena no representa ninguna expresión numérica correcta se comete el error C.

VAL\$ argumento alfanumérico

Esta función es similar a VAL, con la diferencia de que los caracteres de la cadena que hace de argumento deben formar una expresión alfanumérica (cadena) en lugar de una expresión numérica. Verbigracia, VAL\$ "a\$" = "a\$"

VAL\$ (z\$ + "(2,3)") = a\$(2,3) si z\$ = "a\$"
 b\$(2,3) si z\$ = "b\$"

Esta función podría ser útil para seleccionar entre varias opciones una matriz bidimensional de caracteres. Se cae en el error C si la cadena no representa una expresión alfanumérica correcta.

VERIFY etc

Sentencia empleada para releer la información grabada en una cinta cassette (Sección 9) o en otro dispositivo periférico (10.5) y compararla con la información que hay en el ordenador. Sirve para comprobar grabaciones (SAVE).

5. Expresiones

Una expresión es cualquier agrupación de números, cadenas, variables, funciones y operadores combinados para producir un número o una cadena. Algunos ejemplos de expresiones numéricas son:

2

potaje ↑ 2.6e-5

(a(3,huevos/7) + VAL\$(porque TO 15)) *9.8

tiempo > = 100 OR INKEY\$ < > "" "" ""

y como expresiones alfanuméricas podemos citar:

“trenza”

CHR\$ 114 + “operacion”(1 TO 12/4)

p\$ AND hilo = 0

Como regla general aplicable al BSIC Spectrum, se puede decir que no existe la obligación de utilizar números o cadenas simples, siendo posible recurrir a expresiones en su lugar; por ejemplo, GO TO 100 *x.

Dos excepciones de esta regla son el número de línea que hay siempre al comienzo de cada línea, y las partes de un número expresado en notación exponencial o científica (con ‘e’), es decir, no es válido escribir ‘2 *a PRINT 9’ como línea de programa, ni ‘PIe(x + 8)’ como número.

5.1. Números

El ordenador Spectrum almacena entre 9 y 10 dígitos de un número (lo codifica en sistema binario, en el cual el décimo dígito puede introducir un pequeño error). No importa que los dígitos estén antes o después de la coma (punto) decimal, es decir, tanto 12345678900 como 0.000123456789 se almacenan con toda precisión.

El mayor número con el que puede trabajar el Spectrum es 10^{38} aproximadamente ($1e38$), y si un cálculo da lugar a un número mayor que éste, o bien a un número negativo menor que $-1e38$, se incurre en el error 6.

El menor número positivo admisible es, aproximadamente, 4×10^{-39} ($4e-39$), y si algún cálculo da como resultado un valor comprendido entre dicho número y $-4e-39$, el ordenador lo considera igual a 0.

Los números pueden escribirse con o sin coma decimal, y pueden ir acompañados, opcionalmente, por un *exponente* para indicar que debe multiplicarse por una potencia de 10; el exponente consta de ‘e’ o ‘E’, un signo + opcional o un signo - obligatorio, y de un número carente de coma y exponente; por ejemplo, $2e5 = 200000$, $2.3e-6 = .0000023$.

Los números pueden escribirse también en notación binaria, como por ejemplo, BIN 10010 = 18 (consulte la palabra clave BIN en el glosario).

NOTA: A causa de un defecto en la memoria ROM, el número -65536 puede originar resultados incorrectos,

especialmente si se le aplica la función INT o si se produce como resultado de una adición o sustracción, y poco puede hacerse para evitarlo.

5.2. Cadenas

Las cadenas pueden ser todo lo largas que se desee y pueden contener cualquier carácter. Se escriben entre comillas, como por ejemplo, "cafetera".

Las comillas consideradas en sí mismas como un carácter, pueden utilizarse en una cadena, pero en tal caso han de repetirse para indicar que no se trata del final de la cadena. Por ejemplo: """"Hurra!""""grito el." La cadena se almacena en el ordenador sin repeticiones.

Entre las funciones que pueden aplicarse a las cadenas están la suma, +, las comparaciones: >, =, etc., AND (y), y la *fragmentación*, una operación que consiste en seleccionar parte de una cadena (véase el apartado 5.5).

5.3. Variables

Existen diversos tipos de variables: variables simples para números o cadenas; matrices numéricas y matrices de cadenas (alfanuméricas), y los argumentos de DEF FN. En general, las variables se identifican con una sola letra como nombre, seguida por el símbolo \$ en el caso de cadenas y matrices de caracteres. No obstante, las variables numéricas simples (no matrices) pueden identificarse por medio de una letra seguida de cualquier combinación de letras y dígitos. Por ejemplo:

Números: x, y2, autobus numero 53

Cadenas: a\$, b\$, etc.

Matrices numéricas: a(...), b(...), etc.

Matrices de caracteres: a\$(...), b\$(...), etc.

Argumentos de DEF FN: a, b para números

a\$, b\$ para cadenas

Los espacios que haya dentro de los nombres se ignoran, igual que los caracteres de control de colores. Asimismo, las letras mayúsculas y minúsculas se consideran equivalentes, de modo que 'Auto bUsn UME ro5 3' se considera la misma variable que 'autobus numero 53'.

Una variable numérica simple y una matriz numérica pueden compartir la misma letra como nombre, aunque sean ellas mismas distintas. O sea, es perfectamente válido escribir:

```
10 LET a = 19: DIM a(5)
```

Ahora tendremos una matriz 'a' y una variable 'a'. Por el contrario, una variable alfanumérica no puede compartir el nombre con una matriz de caracteres, porque al establecer la dimensión (DIM) de la matriz se borra la variable.

El argumento de una función tipo DEF FN puede tener el mismo nombre que una variable ordinaria, la cual se olvida temporalmente, mientras se está evaluando la función FN. Por ejemplo:

```
10 DEF FN c$(x$) = x$ + y$  
20 LET x$ = "opt": LET y$ = "imista"  
30 PRINT FN c$("pes"): REM escribe "pesimista"
```

Las variables se establecen por medio de sentencias LET (también INPUT, READ o FOR) las simples, sentencias DIM para las matrices, o de forma provisional —por medio de FN para argumentos de DEF FN—. Hasta entonces la variable no existe, y si se incluye en una expresión se incurre en el error 2. Recuerde que CLEAR y RUN borran todas las variables.

5.4. Matrices

Las matrices numéricas son absolutamente convencionales; se admite cualquier número de dimensiones, estando su tamaño limitado únicamente por el espacio disponible en la memoria. El nombre debe estar formado por una sola letra. Por otro lado, una sentencia DIM olvida cualquier otra matriz anterior que tuviera el mismo nombre (por lo cual es importante asegurarse de que tal sentencia DIM se ejecuta únicamente una vez en el programa) y establece una nueva con todos sus elementos inicializados con el valor 0.

Los subíndices empiezan por 1 y llegan a un valor igual a la dimensión de la matriz. No se admite el 0 como subíndice. Resumiendo, después de haber establecido, por ejemplo, DIM a(6,8), tanto a(0,2) como a(7,2) provocan error 3 (la segunda porque el subíndice 7 supera la dimensión, 6).

En sus aplicaciones más simples, las matrices de caracteres son iguales, con la salvedad de que almacenan caracteres en lugar de números, inicializándolos como espacios. El nombre consta de una letra seguida del signo \$, y en caso de que ya hubiera otra matriz con el mismo nombre sería borrada. .

Si bien el programa puede modificar el número de caracteres almacenados en una variable alfanumérica simplemente con una nueva sentencia LET, en las matrices alfanuméricas el número de caracteres es fijo, y está determinado por la sentencia DIM. Esto se pone de manifiesto cuando se asigna un valor a un elemento de una matriz de caracteres con una sentencia LET: la parte derecha de la expresión puede ser una combinación de caracteres cualquiera, pero sólo se toma el primero. Es decir:

DIM a\$(3,5):LET a\$(1,1) = "abcde"

a\$(1,1) adoptará el valor "a", pero el resto de los elementos de la matriz seguirán siendo espacios.

Se puede omitir el último subíndice (no así los demás) en una matriz de caracteres, de modo que en lugar de obtener un solo carácter se obtiene una cadena en la que se han considerado todos los valores posibles del subíndice omitido; por ejemplo, siendo a\$ la del ejemplo anterior, a\$(2) es la cadena compuesta por a\$(2,1), a\$(2,2) ..., a\$(2,5). Así a\$ parece ser una matriz compuesta por 3 cadenas de longitud fija, a\$(1), a\$(2) y a\$(3).

Como a\$(2) tiene una longitud fija igual a 5, si se le asignan caracteres en exceso, los que sobren serán ignorados, o sea, después de escribir

LET a\$(2) = "tablado"

a\$(2) tiene el valor "tabla" (esto significa que a\$(2,1) = "t", a\$(2,2) = "a", ... etc., hasta a\$(2,5), que vale "a"), puesto que sólo se necesitan 5 caracteres.

Si el valor de la parte derecha tiene menos caracteres de los 52 necesarios —como, por ejemplo, LET a\$(2) = "uva"— se completará el valor con espacios: a\$(2) valdrá "uva ", con dos espacios al final.

En el caso de una matriz de caracteres unidimensional, se puede prescindir no sólo del subíndice, sino también de los paréntesis: así se asemeja bastante a una variable alfanumérica simple, aunque de longitud fija; por ejemplo:

DIM b\$(7):LET b\$ = "diplodocus":PRINT b\$

haría aparecer en la pantalla "diplodo".

Consulte también la sección siguiente, que tiene que ver con las matrices de caracteres.

5.5. Fragmentación de cadenas

La *fragmentación* es un procedimiento empleado para seleccionar parte de una cadena por medio de una notación que recuerda a los subíndices de las matrices. Por ejemplo:

"azafran"(5) = "r"	(el quinto carácter)
"bicicleta"(2 TO 4) = "ici"	(del segundo al cuarto carácter)

este método no es el convencional de BASIC, sustituye a las funciones LEFT\$, RIGHT\$ y MID\$. Se puede tomar un solo carácter o, recurriendo a TO, una sección entera. Son varias las posibilidades:

(posición)	(un sólo carácter)
(comienzo TO final)	(una sección)
(TO final)	(una parte que empieza por el principio de la cadena; es decir, 'comienzo' = 1)
(comienzo TO)	(una sección que llega hasta el final; es decir, 'final' = longitud de la cadena)
(TO)	toda la cadena (puede utilizarse para fijar la longitud de una variable alfanumérica simple)
()	—véase el comentario siguiente—)

Normalmente, si los valores 'posición', 'comienzo' o 'final' no están comprendidos entre 1 y la longitud de la cadena, se produce el error 3, pero con el formato TO no se aplica esta regla, a condición de que el final esté antes que el principio (normalmente esto no ocurre); en tal caso, 'final' puede valer incluso 0, y entonces el resultado es la cadena vacía "". Por ejemplo:

"gas" (0 TO 4) da error 3 (0 es demasiado pequeño, y 4 demasiado grande)

“gas” (4 TO 0) = “”

Esta posibilidad nos permite varios trucos, tales como definir una función para producir la ‘cola’ de una cadena (todos sus caracteres menos el primero):

10 DEF FN c\$(x\$) = x\$(2 TO)

Con la regla anterior, esta sentencia puede funcionar cuando x\$ sólo tiene un carácter o ninguno en absoluto.

La fragmentación puede aplicarse no sólo a cadenas y variables alfanuméricas, sino también a expresiones alfanuméricas encerradas entre paréntesis, tales como

(a\$ + “ganso”)(5 TO 7)

Con las matrices de caracteres, el último subíndice puede sustituirse por un fragmento; verbigracia, si a\$ es bidimensional, a\$(2,2 TO 4) = a\$(2)(2 TO 4), y está formada por 3 caracteres, a\$(2,2), a\$(2,3) y a\$(2,4).

Cuando las variables alfanuméricas y las matrices de caracteres son fragmentadas en la parte izquierda de una sentencia LET, los caracteres de la sección fragmentada son asignados respetando una longitud fija, mientras que los demás permanecen inalterables; por ejemplo:

LET a\$ = “brasa”:LET a\$(2 TO 3) = “lucro”

Ahora, a\$ = “blusa” (“cro” fue eliminada en la asignación de longitud fija).

5.6. Operadores y funciones

Los operadores que contiene el Spectrum son:

- ↑ (exponenciación —elevar un número a una potencia—. El primer número no debe ser negativo. Por ejemplo, (-3) ↑ 2 da error A)
- *, /
- +, -, =, <, >, <=, >=, <> (comparaciones —véase la sección siguiente—. Los dobles se teclean con SYMBOL SHIFT y Q, W o E)
- AND (véase la sección siguiente)
- OR (véase la sección siguiente)

Se han enumerado por orden de prioridad, el cual determina, cuando no existen paréntesis, el orden en el que serán ejecutadas las operaciones. $\uparrow$ tiene la prioridad más alta, $*$ y $/$ van en segundo lugar, con la misma prioridad ambas, y así hasta llegar a OR, que tiene la prioridad más baja. Las operaciones de mayor prioridad se ejecutan en primer lugar, y las operaciones de igual prioridad se ejecutan de izquierda a derecha —a menos que se indique otra cosa por medio de paréntesis. Por ejemplo:

$$1 + 2 * 3 \uparrow 4/5 = 1 + ((2 * (3 \uparrow 4))/5)$$

$+$, las comparaciones, y AND (como cadena AND número) pueden utilizarse también con cadenas. La suma de dos cadenas es el resultado de unir las una al final de otra, como por ejemplo:

“boca” + “manga” = “bocamanga”

En cuanto a las versiones alfanuméricas de las demás operaciones, consulte la sección siguiente.

Hay varias funciones internas disponibles, tales como SIN, COS, LEN, CHR\$, etc., las cuales también tienen un nivel de prioridad asignado, mayor que el de $\uparrow$, de manera que no es necesario encerrarlas entre paréntesis donde se apliquen (a su argumento), a menos que contengan uno de los operadores enumerados anteriormente. Es decir

Se puede escribir SIN x	por SIN(x)
Se puede escribir SIN 33	por SIN(33)
Se puede escribir ASN SIN a(2,4)	por ASN (SIN(a(2,4)))
Se puede escribir LEN “elefante”(3 TO)	por LEN(“elefante”(3 TO))
Se puede escribir SIN PI/2 o bien (SIN PI)/2	por SIN. (PI)/2
Pero debe escribir SIN(PI/2)	por SIN(PI/2)
Se puede escribir VAL\$ a\$(3 TO 6)	por (VAL\$(a\$))(3 TO 6)

Observe que $\text{VAL}\$(a\$(3 \text{ TO } 6)) = \text{VAL}\$(a\$(3 \text{ TO } 6))$

Las funciones son:

Argumento numérico, resultado numérico. SIN, COS, TAN, ASN, ACS, ATN, EXP, LN, SQR, INT, SGN, ABS, PEEK, IN, USR.

Argumento alfanumérico, resultado numérico. CODE, VAL, LEN, USR.

Argumento numérico, resultado alfanumérico. STR\$, CHR\$.

Argumento alfanumérico, resultado alfanumérico. VAL\$.

Las funciones siguientes funcionan de manera ligeramente distinta.

+ y - cuando se utilizan únicamente con un número a la derecha, como por ejemplo, +5, -(x+9.1), tienen la misma prioridad que el + y el - ordinarios; verbigracia, $-x \uparrow 2 = -(x \uparrow 2)$.

NOT (ver la siguiente sección) tiene un nivel de prioridad intermedio entre el de las comparaciones y el de AND, de modo que su argumento debe ir entre paréntesis sólo si contiene algún operador AND u OR, o sea:

NOT puntos/2 > = 1000 AND INKEY\$ = "P"
= (NOT((puntos/2) > = 1000)) AND (INKEY\$ = "P")

RND, INKEY\$ y PI pueden considerarse como funciones que no necesitan argumento.

INKEY\$ # es, en realidad, una función de un argumento, el número de corriente. Teclee INKEY\$ y # por separado.

SCREEN\$, ATTR y POINT tienen dos argumentos, que van entre paréntesis. Por ejemplo: SCREEN\$(10,16).

BIN tiene todo el aspecto de una función, pero en realidad es sólo un símbolo de notación, pues su 'argumento' sólo puede constar de ceros y unos.

Funciones definidas por el usuario. Véase el apartado 5.8. Deben llevar siempre paréntesis, como en FNa(2) o FNp().

5.7. Operadores lógicos

Las comparaciones =, <, >, <=, >=, y <>, así como los operadores lógicos AND, OR y NOT, se utilizan

principalmente en sentencias IF para definir la condición cuyo cumplimiento se investiga. En esta sección se estudia en primer lugar su aplicación más simple, para entrar después en detalles técnicos.

La forma más elemental de considerar estos operadores, válida para la mayoría de los casos, es pensar que las comparaciones producen el resultado 'cierto' ('sí, la comparación se cumple') o 'falso' ('no, no se cumple'). Ejemplos: ' $2 = 2$ ', ' $9 < 10$ ', ' $9 \leq 10$ ', ' $9 \leq 9$ ' y ' $-3 < -2$ ' son todas ciertas, mientras que ' $2 = 3$ ', ' $10 < 9$ ', y ' $9 < 9$ ' son falsas. Estos valores 'ciertos' en abstracto son una mera interpretación humana: el ordenador Spectrum utiliza en realidad los números 1 y 0.

Cuando la comparación se incluye en una sentencia IF, como

`IF a < > 0 THEN PRINT a`

la instrucción introducida por THEN será ejecutada si la comparación se cumple (es cierta).

Habiendo utilizado las comparaciones para producir valores de verdad, se pueden combinar éstos usando NOT, AND y OR.

NOT invierte un valor, es decir, cambia cierto por falso y viceversa. Por ejemplo, 'NOT $2 = 2$ ' es falso (puesto que ' $2 = 2$ ' es cierto). Es otra forma de escribir ' $2 < > 2$ '.

AND combina dos valores, siendo el resultado cierto si ambos valores lo son. Por ejemplo:

' $8 < 9$ AND $2 < 100$ ' es cierto

' $8 < 9$ AND $2 > 100$ ' es falso, porque ' $2 > 100$ ' es falso.

Si se combinan más de dos valores con AND, como, por ejemplo, en:

`tiempo <= 20 AND a$ = "si" AND
SCREEN$(linea,0) < > " "`

entonces, todos ellos deben ser ciertos para que lo sea la combinación.

OR sirve para combinar dos valores de verdad de forma tal que la combinación es cierta si uno al menos de los valores originales lo es (o posiblemente ambos). Ejemplos:

'8 < 9 OR 2 < 100' es cierto
'8 < 9 OR 2 > 100' es cierto (porque '8 < 9' es cierto)
'8 > 9 OR 2 > 100' es falso

Si se combinan diversos valores de verdad con OR, como en

```
x > 22 OR k$ = "n" OR k$ = "N" OR k$ = " " OR  
k$ = "STOP"
```

entonces, por lo menos uno de ellos debe ser cierto para que lo sea la combinación.

Se pueden construir combinaciones más complicadas, utilizando paréntesis si fuese necesario. Por ejemplo:

```
IF x > 0 AND (NOT d$ = "no" OR INKEY$ = "p")  
THEN PRINT x
```

Es importante recordar el orden de prioridades —en primer lugar las comparaciones, luego NOT, después AND y por último OR— porque una lectura superficial puede dar una impresión equivocada. En el ejemplo anterior, NOT se aplica únicamente a 'd\$ = "no"' (siendo el resultado equivalente a 'd\$ < > "no"') y no a toda la estructura 'd\$ = "no" OR INKEY\$ = "p"' (como podría pensarse). Si ponemos paréntesis en todos los lugares tendremos

```
(x > 0) AND ((NOT (d$ = "no")) OR (INKEY$ = "p"))
```

Resumen de las aplicaciones simples comunes a todas las versiones de BASIC

Los operadores de comparación =, >, <, >=, <=, y <> comparan números o cadenas dando como resultado los valores 'cierto' o 'falso'.

NOT, AND y OR operan con valores de verdad ('cierto' o 'falso').

IF es cierto o falso, y según el caso, se ejecuta la parte THEN ('cierto') o no se ejecuta ('falso').

Nota: Las cadenas se comparan según un criterio de orden alfabético basado en el conjunto de los 256 caracteres clasificados en función de sus códigos numéricos. Para que la comparación sea cierta, la primera cadena debe estar antes según este peculiar orden alfabético. Por ejemplo:

"huevos" < "jamón" y "pan" <= "pan" son ciertas

“huevos” > “jamón” y “pan” < “pan” son falsas

Una particularidad de este orden es que todas las mayúsculas vienen antes que todas las minúsculas, de modo que “Solo” < “sol” es cierto, aunque en cualquier diccionario, ‘sol’ estaría antes.

Características básicas. Los siguientes detalles son particulares del BASIC Spectrum.

El Spectrum considera los resultados de las comparaciones como números, no como valores abstractos ‘cierto’ o ‘falso’: 1 corresponde a cierto, y 0 a falso. En consecuencia, los resultados de las comparaciones pueden almacenarse en variables, de modo que es lícito escribir:

```
100 LET x en margen = 10 <= x AND x <= 21
110 IF x en margen AND INKEY$ = "y" THEN PRINT x
```

(No confunda el signo = de la sentencia LET con el operador de comparación =)

IF puede tomar cualquier número por condición, aun cuando normalmente sea 0 ó 1. IF considera 0 como ‘falso’ y todos los demás números (incluyendo el 1) como ‘cierto’.

En realidad, los operadores lógicos también trabajan con números. Su definición completa está concebida para dar las respuestas que se espera de ellos, con los valores de verdad 0 y 1 (es un ejercicio muy instructivo comprobarlo).

NOT siempre da 0 ó 1 como resultado: 1 si su argumento es 0, y 0 si su argumento toma cualquier otro valor. Por ejemplo, NOT 0 es 1, mientras que NOT 1, NOT 0.126 y NOT -10024 son todos 0.

AND aplicado a números que no sean 0 ó 1 se comporta de forma distinta con los dos números que combina. El segundo es una condición que determina si se va a tomar el valor del primero o (en su lugar) 0. En efecto:

$$x \text{ AND } y = \begin{cases} 0 & \text{si } y = 0 \\ x & \text{si } y \neq 0 \end{cases}$$

En este último caso, AND se lee como “si”, que define mejor su significado. Esta característica resulta útil para construir expresiones condicionales, tales como

$$(9.1 \text{ AND } a = 9) + (8.9 \text{ AND } a = 8)$$

cuyo valor es 9.1 si $a = 9$, 8.9 si $a = 8$, y 0 en cualquier otro caso.

El mismo principio puede aplicarse a cadenas:

$$x\$ \text{ AND } y = \begin{cases} \text{"si"} & \text{si } y = 0 \\ x\$ & \text{si } y < > 0 \end{cases}$$

Observe que $x\$$ es una expresión alfanumérica, pero que y es numérica —en general responde a una condición de cualquier clase, como

`PRINT ("si" AND r$ = "s") + ("no" AND r$ = "n");`

cuyo resultado sería escribir "si" o "no" según que $r\$$ sea "s" o "n".

OR se define como

$$x \text{ OR } y = \begin{cases} x & \text{si } y = 0 \\ 1 & \text{si } y < > 0 \end{cases}$$

(Léase OR como 'a menos que')

Esta característica sirve también para construir expresiones condicionales, tales como

$$(9.1 \text{ OR } a < > 9) * (8.9 \text{ OR } a < > 8)$$

aunque es menos útil que AND. No existe versión de OR para cadenas.

5.8. Funciones definidas por el usuario

El usuario puede definir hasta 26 funciones numéricas (identificadas por FN y una letra) y otras 26 funciones alfanuméricas (FN, una letra y el símbolo \$) con sentencias DEF FN. Siempre que en algún lugar del programa haya una sentencia DEF FN, la correspondiente función FN puede utilizarse en cualquier parte del mismo, incluso en líneas de comando. Cada función puede tener hasta 26 argumentos alfanuméricos (cuyo nombre es una sola letra) y 26 argumentos alfanuméricos (identificados por una letra acompañada del signo \$). Por ejemplo:

```

10 DEF FN m(x,y) = (x AND x >= y) + (y AND
x < y):REM el mayor de x e y
20 DEF FN r$(x$,y) = x$(FN m(1,LEN x$ - y + 1)
TO):REM los y caracteres situados a la derecha de x$
30 DEF FNd() = INT (RND *6) + 1:REM jugada arbitraria
de un dado

```

Cuando se invoca una función, los argumentos reciben los valores indicados entre paréntesis (sustituyendo temporalmente a otras variables que tengan el mismo nombre) y se evalúa la expresión de la sentencia DEF FN. Incluso si no hubiera argumentos, los paréntesis siguen siendo necesarios, como en FN d().

El programa será ejecutado con más rapidez si las sentencias DEF FN se colocan cerca del principio.

No se permiten recurrencias; en otras palabras, en la definición de una función no puede aparecer la misma función, ya sea directamente, como en

```
10 DEF FN a(x) = FN a(x + 1) *2
```

o indirectamente, a través de alguna otra función. Si se intenta hacerlo, se producirá normalmente el error 4, pero en ocasiones fallará todo el sistema.

Otros errores que pueden cometerse son: P si se utiliza una función FN que no haya sido definida en una sentencia DEF FN en el programa; Q si los argumentos de FN no coinciden exactamente con los de la definición DEF FN. Por ejemplo, a FN m(5) le faltan argumentos, y FN r\$("gato", "perro") tiene un argumento alfanumérico, "perro" que debería ser numérico según la definición.

6. Estructuras

Existen relativamente pocos procedimientos para estructurar un programa en BASIC Spectrum: líneas multisentencia, IF...THEN, FOR...NEXT, y GO SUB...RETURN.

6.1. Líneas multisentencia

Se pueden incluir varias sentencias en una sola línea, separadas por dos puntos (:). Por ejemplo:

```
FOR f = 32 TO 255:PRINT CHR$ f:NEXT f
```

No es posible utilizar GO TO para saltar a una sentencia situada en medio de una de tales líneas, y las sentencias REM únicamente pueden colocarse al final de la línea.

6.2. IF...THEN

La sintaxis es

IF condición THEN sentencias

por ejemplo:

```
50 IF a = 2 THEN PRINT x:LET x = x + 1: GO TO 20
```

Lo que hemos llamado 'condición' es, técnicamente, una expresión numérica, pero en la práctica se trata de una comparación (véase el apartado 5.7). Las sentencias introducidas por THEN constituyen el resto de la línea, separadas de la manera habitual, con dos puntos. En el ejemplo propuesto, si $a < 2$ no se ejecuta ninguna sentencia de la línea.

No es válido el formato 'IF $x = 5$ THEN 200', que no es admisible, hay que escribir 'IF $x = 5$ THEN GO TO 200'.

6.3. FOR...NEXT

FOR variable de control = valor inicial TO valor final
o bien

FOR variable de control = valor inicial TO valor final STEP
paso

NEXT variable de control

'valor inicial', 'valor final' y 'paso' son expresiones numéricas.

Las únicas variables que pueden utilizarse como variables de control son variables numéricas simples cuyos nombres están compuestos por una sola letra; por ejemplo:

```
FOR f = 1 TO 10
```

```
FOR n = 50 TO 20 STEP -PI/2
```

son válidas, mientras que

```
FOR cuenta = 1 TO 10  
FOR a(1,2) = 1 TO 10
```

no lo son.

Las expresiones 'valor inicial' y 'valor final' no son reevaluadas cada vez que se ejecuta el bucle; o sea, en

```
10 LET a = 1  
20 FOR f = 1 TO 10 STEP a  
30 LET a = a + 1:PRINT f  
40 NEXT f
```

el valor del 'paso' (step) es siempre 1, y no aumenta junto con a.

Al final del bucle, la variable de control tiene un valor un paso mayor que el que tenía en la última ejecución del bucle (o sea, $f = 11$ al final del último ejemplo), porque NEXT la ha hecho variar una vez más.

Si el valor inicial es superior al final, como sucede en `FOR f = 10 TO 1`, o en `FOR g = 1 TO 5 STEP -1`, el bucle no será ejecutado ninguna vez. El ordenador buscará el NEXT correspondiente en todo el programa (si no existe se produce un error I), y procederá a partir del mismo, dejando la variable de control con el valor inicial.

Se puede salir de un bucle FOR sin problemas, y volver al mismo más tarde en caso necesario, pero de otro modo hay que entrar en un bucle FOR únicamente a través de la sentencia FOR.

Observe que cuando se especifica un valor de paso no entero, el error de redondeo puede hacer que en lugar de alcanzar el valor final exacto, la variable de control lo sobrepase ligeramente y acabe de forma prematura. Por ejemplo, en `'FOR n = 0 TO 1 STEP 0.1'`, n toma los valores 0, 0.1, 0.2, ..., 0.9, pero no llega a 1. La mejor forma de solventar esta dificultad es hacer el valor final un poco mayor de lo necesario, tal como, para el último ejemplo, `FOR n = 0 TO 1.01 STEP 0.1`

6.4. GO SUB y RETURN

Ejemplo:

```
50 LET s = 5:GO SUB 1000  
1000 REM imprimir s espacios
```

```
1010 FOR i = 1 TO s:PRINT " ";:NEXT i
1020 RETURN
```

Esta es la forma BASIC estándar. La recurrencia (es decir, una rutina GO SUB se llama a sí misma) es técnicamente posible, pero se recomienda evitarla.

Como el número de línea que va después de GO SUB puede ser cualquier expresión numérica, se pueden hacer cosas como

```
5 LET espacios = 1000
10 LET s = 5:GO SUB espacios
```

Cuando hay más GOSUBs que RETURNs la recurrencia provoca un error 4, que se puede borrar con CLEAR.

7. Color y gráficos

En esta sección nos dedicaremos a estudiar la formación de la imagen de TV y la utilización de gráficos. En la Sección 8 se estudia más detalladamente la utilización de la imagen por parte de las sentencias PRINT.

7.1. Imagen-pixels y atributos

La imagen de TV es un rectángulo de puntos, denominados pixels (elementos de imagen). Concretamente, tiene 256 pixels de ancho por 192 de alto. Una imagen en particular se forma especificando el color de cada pixel de la pantalla.

Un carácter representado en la pantalla ocupa un cuadrado de 8 por 8 pixels, conocido como *posición de carácter*. Así pues, toda la pantalla tiene 32 posiciones de caracteres de ancho (32 columnas) y 24 caracteres de alto (24 líneas). Cuando PRINT coloca un carácter en la pantalla, lo que hace es definir un cuadrado particular de 8 por 8 pixels.

No se tiene libertad absoluta para determinar el color de todos y cada uno de los pixels, porque dentro de una posición de carácter dada, sólo puede haber dos colores a un tiempo, que se denominan *color de tinta* y *color de papel*, dado que PRINT emplea el color de tinta para el carácter propiamente dicho, y el color de papel para el fondo. Existen 8 colores entre los que se puede escoger, cada uno identificado por un código numérico

comprendido entre 0 y 7 (se encuentran sobre las teclas de los dígitos correspondientes en el teclado).

Así pues, para cada pixel el ordenador sabe si debe tener el color de tinta o de papel (tomándolo de la posición de carácter que ocupa), y para cada posición de carácter, el ordenador conoce los respectivos colores de tinta y papel. El usuario tiene la posibilidad de resaltar cualquier posición (haciendo más brillantes sus dos colores) o de hacerla intermitente (alternando los dos colores a razón de unas tres veces por segundo).

Estas cuatro propiedades, los dos colores y las opciones de brillo e intermitencia, se conocen como los *atributos* de la posición del carácter, y se aplican a los 64 pixels que la componen. En la sección siguiente se enseña a utilizar los pixels y sus atributos.

Puede seleccionarse uno de los 8 colores para la superficie que rodea a la zona de la pantalla de TV utilizada por el ordenador, por medio de la palabra clave **BORDER**.

Para especificar un pixel o una posición de carácter se necesitan dos números, llamados *coordenadas*.

Para una posición de carácter, la primera coordenada es el número de línea (desde 0, correspondiente a la línea superior de la pantalla, hasta 23, que es la última), y la segunda indica el número de columna (0 a la izquierda y 31 a la derecha). Estas coordenadas se utilizan en **AT**, **SCREEN\$** y **ATTR**; por ejemplo:

```
PRINT AT 0,10;"Titulo"
```

Imprimirá la palabra "Titulo" empezando en la décima columna de la línea superior.

En cuanto a un pixel, la primera coordenada —o coordenada *x*— indica la distancia que lo separa del borde izquierdo, que puede ser de 0 a 255. La segunda —o coordenada *y*— es la distancia medida desde el borde inferior; considerando que no se pueden emplear las dos líneas inferiores para gráficos (quedando, por consiguiente, excluidas de este sistema de coordenadas), los posibles valores de la coordenada *y* están comprendidos entre 0 y 175 (en lugar de 191). Estas coordenadas se emplean en **POINT**, y en las sentencias de construcción de gráficos **PLOT** y **CIRCLE**.

Hay tres funciones cuya finalidad es inspeccionar la imagen visualizada:

POINT indica si un pixel pertenece a color de tinta o de papel.

ATTR comunica los 4 atributos de una posición de carácter determinada, codificándolos en un único número.

SCREEN\$ examina los pixels de una posición de carácter dada y averigua el carácter que forman, si acaso forman alguno.

7.2. Combinaciones de colores

Todo lo que afecte a la pantalla tendrá que ajustar no sólo los pixels, sino también los atributos de su correspondiente posición de carácter. Por ejemplo, PRINT INK 2;"tomate", además de determinar los pixels pertinentes para formar letra a letra la palabra "tomate" en 6 posiciones de carácter sucesivas, establece el color de tinta en rojo, para que "tomate" se escriba en dicho color. La tinta (escritura) roja sólo se mantiene en este caso hasta el final de la sentencia PRINT, pero también es posible hacer de INK 2 una sentencia tal que todas las sentencias PRINT utilicen tinta roja.

Hay varias posibilidades de control de pixels y atributos por parte de PRINT y otros tipos de sentencias de impresión, las cuales se denominan combinaciones de colores, aunque de hecho influyen en otras características además de los colores. En términos generales, dan instrucciones a PRINT para ajustar un atributo a un valor definido o para dejarlo tal como está, aunque pueden producirse diversos efectos visuales. Una combinación de color queda definida por seis números, los llamados valores de INK (tinta), PAPER (papel), FLASH (intermitencia), BRIGHT (brillo), OVER (características de sobreimpresión) e INVERSE (inversión de color papel-tinta). Obsérvelos bajo la fila inferior del teclado. Por ejemplo, nada más activar el Spectrum, la combinación de color que aparece se caracteriza por los siguientes valores: INK 0 (negro), PAPER 7 (blanco), FLASH 0 (fijo), BRIGHT 0 (caracteres sin destacar), OVER 0 e INVERSE 0 (más adelante explicaremos estas dos últimas características). En conjunto, todo ello significa que la escritura es negra sobre fondo blanco, fija y sin brillo especial.

Valores de INK y PAPER. Estos valores controlan los atributos INK y PAPER, y se encuentran dentro del margen de

0 a 9. Un valor comprendido entre 0 y 7 significa un color definido, e indica a PRINT que ajuste los atributos a este color. El valor 8 ('transparente') implica dejar el atributo invariable en la pantalla, y el valor 9 ('contraste') significa cambiar el atributo a blanco o negro de forma que contraste con el otro (el negro contrasta con los colores claros: verde, cyan, amarillo y blanco, mientras que el blanco, a su vez, contrasta con los otros cuatro colores, los más oscuros). Por ejemplo, PAPER 2, INK 9 significa que el color de papel será rojo en cualquier sitio donde se realice una impresión, y el color de tinta se hace blanco para contrastar con el rojo.

PAPER 9, INK 8 significa que la impresión no altera el color de la tinta, pero cambia el del papel para lograr que contraste con aquél.

Con PAPER 9, INK 9, en primer lugar se hace que el color de papel contraste con el antiguo color de tinta, y luego se hace que el nuevo color de tinta contraste con el nuevo del papel.

Valores de BRIGHT y FLASH. Estos valores controlan los atributos de intermitencia y brillo. 0 y 1 significan desactivar y activar el atributo, respectivamente (0 = desactivado, o sea, sin intermitencia o con brillo normal; 1 = activado, es decir, intermitente o brillante). 8 significa dejarlo tal como está.

Valores de OVER e INVERSE. Estos valores no afectan a los atributos, sino a los pixels.

OVER: normalmente (OVER 0) imprime papel sobre todos los pixels antes de imprimir la nueva imagen, pero con OVER 1, en lugar de borrar lo que hubiera antes, se combina con lo nuevo de acuerdo con las reglas siguientes:

papel + tinta = tinta
y papel = papel = tinta + tinta = papel

Es decir, un nuevo pixel de tinta borra el antiguo. Esto significa que, con OVER 1, si se imprime lo mismo dos veces en el mismo lugar, se obtienen la misma combinación de pixels con el que empezó.

INVERSE. INVERSE 0 significa impresión normal, con tinta para el carácter propiamente dicho, y papel para el fondo. INVERSE 1 invierte esta situación.

Resumen de valores posibles:

INK, PAPER	de 0 a 9
FLASH, BRIGHT	0, 1 u 8
OVER, INVERSE	0 ó 1

Hay tres combinaciones de colores. Dos de ellas, las de las partes superior e inferior de la pantalla, se llaman permanentes, mientras que la tercera es temporal.

La combinación permanente para la parte superior de la pantalla es, inicialmente, la clásica de negro sobre blanco, y puede modificarse con *sentencias* INK, PAPER, etc.

La combinación permanente de la parte inferior de la pantalla tiene INK 9, PAPER el color del borde, y todos los demás valores son 0. No puede ser modificada, salvo con una *sentencia* BORDER.

La combinación temporal de colores es la única que se utiliza directamente. Al principio de una *sentencia* PRINT (por poner un ejemplo), se copia de la combinación permanente, pero puede ser modificada con carácter temporal mediante *elementos* (o instrucciones) INK, PAPER, etc., incluidos en las *sentencias* PRINT, y también imprimiendo caracteres de control de color (véanse los apartados 3.3 y 8.5).

Ejemplo:

```

10 INK 1: REM ahora el valor INK permanente es azul
20 PRINT "Esto es azul."
30 PRINT INK 2; "Esto es rojo.": REM tinta roja sólo para
    esta sentencia PRINT.
40 REM El color de tinta permanente sigue siendo azul
50 PRINT "Esto es otra vez azul."

```

Nota: Si se utilizan periféricos con Interface 1 ZX, es posible que se necesite una *sentencia* 'PRINT;' antes de las *sentencias* de color (ver apartado 10.4).

7.3. Gráficos

Las *sentencias* de gráficos PLOT, DRAW y CIRCLE, sirven para dibujar diversas figuras, generalmente pintando pixels, aunque, como veremos a continuación, el procedimiento puede variar.

PLOT dibuja con tinta un único pixel.

DRAW dibuja una línea recta o un arco de circunferencia.

CIRCLE traza una circunferencia completa.

Estas sentencias establecen una combinación temporal de colores basada en la permanente, pero utilizando los parámetros **PAPER 8**, **FLASH 8** y **BRIGHT 8**, lo cual significa que las sentencias determinan el color de la tinta de los pixels donde dibujan, pero no afectan al color del papel. Este procedimiento resulta práctico si se están pintando pixels, pero no lo es tanto si se trabaja con **INVERSE 1** quitando la tinta para dejar el color de papel (recuerde que al cambiar el color de tinta de un pixel resultan afectados también los otros 63 pixels de su posición de carácter).

Se puede modificar esta combinación temporal de colores incluyendo instrucciones de control de colores (elementos de color) en la sentencia gráfica (separados por comas o por puntos y comas), como en

DRAW INK 8; PAPER 6; INVERSE 1; 10,20

Cuando están incluidos en una sentencia **PLOT**, **OVER** e **INVERSE** se comportan de la siguiente manera:

OVER 0; INVERSE 0 se pinta el pixel con tinta.

OVER 0; INVERSE 1 se cubre el pixel con el color de papel. Recuerde que el trazado normal determina el color de tinta, dejando inalterable el del papel —lo que posiblemente no coincida con el deseo del usuario.

OVER 1; INVERSE 0 se cambia el pixel de papel a tinta o a la inversa. Este procedimiento se emplea bastante a menudo en la producción de gráficos, porque para borrar el dibujo basta con volver a pasar por encima del mismo (y queda a la vista lo que había debajo).

OVER 1; INVERSE 1 No afecta al pixel, aunque pueden cambiar los atributos.

7.4. Gráficos definidos por el usuario

Existen 21 caracteres gráficos definidos por el usuario, con códigos que van del 144 al 164, e identificados como 'gráfico a', 'gráfico b', ... etc., hasta 'gráfico u'. Se obtienen en modo G pulsando en el teclado la letra correspondiente, de la A a la U.

Si el usuario redefine un carácter gráfico, no se modifica la forma de ninguna figura que ya esté en la pantalla; únicamente las impresiones posteriores utilizarán el nuevo modelo.

B. Impresión en la pantalla de TV o en la impresora ZX, INPUT

Incluimos la sentencia INPUT en esta sección debido a la gran cantidad de mensajes que pueden hacerse aparecer (en la parte inferior de la pantalla) como advertencias o indicaciones para el usuario ('apuntes'). Todo lo que pueda imprimir PRINT, se puede obtener también con INPUT, de modo que una sentencia INPUT es similar a una sentencia PRINT, pero con la ventaja que supone la posibilidad de introducir variables de paso. Por ejemplo:

```
INPUT "Nombre?";n$;"Numero de cuenta?";cuenta
```

Como se dijo en el apartado 3.1, la pantalla está dividida en dos partes: la superior, para la salida normal, y la inferior, reservada para mensajes informativos y para teclear entradas. Normalmente hay una línea en blanco entre ambas, que se considera perteneciente a la inferior. En general, durante la ejecución de un programa, la parte inferior de la pantalla abarca únicamente las dos líneas de abajo, que no se notan por tener el mismo color del borde; por consiguiente, hay 22 líneas en la parte superior de la pantalla.

Si la impresión alcanza el final de una línea, prosigue automáticamente en la línea que viene a continuación.

Normalmente, PRINT escribe en la parte superior, y cuando llega al final e intenta escribir en la parte inferior de la pantalla, toda la parte superior se desplaza hacia arriba una línea (este efecto se conoce como '*scrolling*'), desapareciendo la primera línea y haciendo sitio para otra línea más. El Spectrum lleva la cuenta de sus desplazamientos ('*scrolls*'), y si descubre que la línea de más arriba (que va a desaparecer de la pantalla) no existía la última vez que se tuvo ocasión de examinar la pantalla (en una sentencia INPUT, o antes de que comenzase la ejecución del programa), se detendrá y formulará la pregunta 'scroll?' (¿desea desplazamiento de la imagen hacia arriba?). Entonces, se puede detener el programa (mensaje D) pulsando

N ('no'), SPACE ('BREAK') o STOP, o dejar que siga su curso pulsando cualquier otra tecla.

La variable del sistema de 1 byte SCR CT (dirección 23692 de la memoria) contiene un valor más que el número de desplazamientos anteriores a la pregunta 'scroll?', de modo que se puede controlar el proceso de 'scrolling' mediante sentencias POKE con las que se cargue el valor deseado en la susodicha dirección. Si lo que desea es que el ordenador ejecute desplazamientos sin preguntar, el método más sencillo consiste en utilizar una sentencia INPUT"" de vez en cuando —así se reajusta el valor de SCR CT para dejar ver que puede desplazarse la totalidad de la parte superior de la pantalla.

La parte inferior de la pantalla es utilizada por sentencias INPUT, y también es posible imprimir en esta zona con PRINT # 1 (véase el apartado 8.2).

8.1. PRINT

Como norma general, una sentencia PRINT contiene un cierto número de *elementos*, separados por puntos y comas (;), comas (,) o apóstrofes (') —se obtiene pulsando simultáneamente las teclas 7 y SYMBOL SHIFT, y provoca un cambio de línea—. Los elementos posibles son expresiones numéricas o alfanuméricas, TAB, AT, elementos de color (INK, PAPER, FLASH, etc.) y el símbolo #. Por ejemplo:

```
PRINT AT 2,0; INK 2; "PI/4 = "; TAB12; PI/4
```

La posición de impresión (o posición PRINT) es el lugar donde aparecerá el próximo carácter.

Las cadenas tienen simplemente sus caracteres uno a continuación de otro.

Para la impresión se convierte un número en una cadena (como sucede con STR\$) a partir de su forma interna. Sólo aparecen ocho dígitos significativos, aunque el ordenador almacena en su memoria hasta 10 dígitos. Si el noveno es 5 o mayor, se redondea el número incrementando por valor de 1 el octavo dígito.

Para los números comprendidos entre 0.00001 y 99999999 se emplea la notación decimal ordinaria, sin exponente, mientras que para los números no comprendidos en tal margen se recurre a la notación exponencial.

Ejemplos:

<i>Número</i>	<i>Se escribe</i>	<i>Notas</i>
10.0	10	No es necesaria la coma (.) decimal
12345678	12345678	
12345678.5	123456789	El octavo dígito se redondea a 9 debido al 5
123456789	1.2345679E+8	Demasiado grande para la notación ordinaria
123.40600	123.406	Se omiten los ceros a la derecha que hay después de la coma decimal
123.406001	123.406	Nunca se escribe el noveno dígito
.123	0.123	Se escribe '0.' cuando el número está entre 0.1 y 1
.0123	.0123	Se escribe '.0' para números menores
.000012345678	.00012345678	
.0000012345678	1.2345678E-6	Demasiado pequeño para la notación ordinaria

Los números negativos se escriben con el signo '-' delante.

En el Spectrum, los números se escriben empezando exactamente en la posición de impresión vigente en el momento (hay algunas versiones de BASIC que imprimen los números de forma que sus últimos dígitos estén alineados en vertical).

Los elementos de color modifican la combinación temporal de colores (véase el apartado 7.2), de forma que su efecto dura únicamente hasta el final de la sentencia PRINT.

El símbolo # se emplea para enviar la impresión a un dispositivo que no sea la pantalla de TV —ver Sección 10—. Una de sus aplicaciones es # 1, para la parte inferior de la pantalla (ver apartado 8.2).

Los tres separadores, a saber, ';', ',' y '"' sirven para controlar la posición de impresión entre elementos:

',' no varía la posición de impresión.

',' imprime entre 1 y 16 espacios, los necesarios para llevar la posición de impresión a la columna 16 de la misma línea, o a la columna 0 de la línea siguiente, con el efecto de tabular la escritura en dos columnas.

“” equivale a un comando ‘newline’ (otra línea) explícito; en efecto, traslada la impresión al principio de la línea siguiente. Al final de una sentencia PRINT se supone que hay un comando ‘newline’ *implícito* (no escrito), a menos que se haya concluido con ‘;’, ‘,’ o ‘.”. ‘newline’ no funciona imprimiendo espacios (como ‘,’), sino que desplaza directamente la posición de impresión.

Si la posición de impresión está ya en el comienzo de una línea, pero sólo porque el último carácter llegó al final de la anterior, el primer ‘newline’ no ejerce efecto alguno. Es decir, PRINT AT 0,31; “x”“y” imprimiría “x” al final de la línea superior, e “y” al principio de la siguiente (de modo que, en apariencia, “” no hace nada).

8.2. Impresión en la parte inferior de la pantalla

Habitualmente, este procedimiento se lleva a cabo como parte de una sentencia INPUT, pero también es posible con PRINT #; su utilidad principal es la generación de mensajes temporales, que serán borrados por INPUT, un mensaje informativo del ordenador, o después de ‘scroll?’. Por ejemplo, para obtener una respuesta tipo “s” (sí) o “n” (no), escribiríamos:

```
10 PRINT # 1; “Desea continuar? (s/n)”
20 PAUSE 0: LET k$ = INKEY$
30 IF NOT (k$ = “s” OR k$ = “n”) THEN GO TO 20
40 INPUT “”: REM para borrar el mensaje; k$ = respuesta
```

La parte inferior de la pantalla también ha sufrido un desplazamiento hacia arriba (*scrolling*), y sin embargo, no se ha perdido la línea de arriba de dicha parte, sino que toda ella se mueve hacia arriba ampliándose con una línea adicional. Puede hacerse hasta obtener un máximo de 23 líneas en la parte inferior de la pantalla; una vez alcanzada esta cifra, cualquier intento ulterior de desplazamiento de la parte inferior produce un error 5. Si en el proceso la parte inferior de la pantalla amenaza con superponerse a la posición de impresión vigente en la parte superior, también será ésta desplazada, desapareciendo su línea superior.

AT funciona de modo diferente en la parte inferior de la pantalla (consulte el Glosario).

La parte inferior de la pantalla tiene su propia posición de impresión, distinta de la de la parte superior.

8.3. INPUT

En el Spectrum, una sentencia INPUT es similar a una sentencia PRINT # que escribiese avisos y mensajes, con la posibilidad adicional de dar entrada a valores para variables. Tiene los mismos elementos (y algunos otros empleados para introducir variables) y los mismos separadores. Por ejemplo:

30 INPUT "Cuanto vale n?",n

(pero no se supone cambio de línea implícito al final).

Todo elemento que empiece por una letra ha de ser una variable de entrada. El programa se detiene entonces, mientras el usuario teclea su valor y pulsa ENTER después. Lo que se teclea aparece en la parte inferior de la pantalla, comenzando por la posición de impresión de esta parte (y luego, dicha posición se desplazará a un lugar situado inmediatamente a la derecha de lo que se ha tecleado), y el usuario puede controlarlo por completo con las teclas de edición y de movimiento del cursor.

Se puede teclear cualquier expresión. Si su sintaxis es incorrecta, aparecerá en la pantalla el signo ? intermitente, dando la oportunidad de editarlo. Como la expresión alfanumérica más habitual es una cadena encerrada entre comillas, el ordenador visualiza automáticamente un par de comillas para las variables de este tipo, pero se pueden borrar si así se desea.

Es posible provocar una interrupción en una sentencia INPUT introduciendo la entrada junto con 'STOP' (teclas SYMBOL SHIFT y A) justo al principio; si se trata de una variable alfanumérica, asegúrese de que STOP vaya antes de las comillas. El resultado es una detención del programa con el mensaje informativo H.

Otro tipo de elemento INPUT es LINE con una variable alfanumérica (ver Glosario). De esta forma se introduce directamente una cadena, sin sus comillas, y sin que sea posible utilizar expresiones alfanuméricas.

Es posible también incluir expresiones en los mensajes; por ejemplo:

50 INPUT "Cuanto vale n ahora?" "El valor antiguo era ";(n)'n

imprimiría el antiguo valor de n como parte del mensaje, pero para indicar que se trata de una expresión a imprimir, y no confundirla con una variable de entrada, debe ponerse entre paréntesis: (n) . Los paréntesis son necesarios para todas las expresiones que comiencen con una letra. Una sentencia tal como 'INPUT $n * 2, x$ ' dará lugar a la aparición del signo ? intermitente, como un error de sintaxis.

Otra característica es la posibilidad de agrupar mensajes entre paréntesis; en este caso se trata, sin lugar a dudas, de mensajes, no de variables de entrada. Por ejemplo:

70 INPUT $(a, b, c), n$

a , b y c son mensajes cuyos valores se imprimen, y n es una variable de entrada. Esto no es que sea demasiado útil, porque siempre puede indicarse por medio de paréntesis individuales, y además puede dar lugar a extraños errores sintácticos. Por ejemplo, la sentencia

INPUT $(a + b) * c, n$

haría aparecer el signo ? intermitente antes de $*$. La solución es poner toda la expresión entre paréntesis, como en

INPUT $((a + b) * c), n$

Otros elementos que pueden incluirse en sentencias INPUT son:

AND y TAB, igual que en PRINT # 1 (véase el apartado 8.2)

Los elementos de color son anulados por una variable de entrada, la cual desactiva la combinación de colores actual y restablece la combinación permanente de la parte inferior de la pantalla.

El signo # incluido en una sentencia INPUT sirve para determinar un dispositivo que no sea la combinación habitual de teclado y parte inferior de la pantalla (véase el apartado 10.4).

INPUT produce dos efectos secundarios: reduce la parte inferior de la pantalla a dos líneas en blanco, y reajusta el valor de la cuenta de scrolling a 23 (evitando la aparición de la

pregunta 'scroll?'). Cuando se desee hacer ambas cosas, resulta útil escribir INPUT "".

8.4. Impresora ZX

Este aparato es una impresora de 32 columnas diseñada para los modelos ZX81 y Spectrum. Aunque la calidad de impresión es realmente mala, el papel metalizado que emplea es relativamente caro, y en conjunto es poco fiable, gracias a su ingeniosa construcción es barata, y si no se le exige demasiado, funciona razonablemente bien. Si desea mejor calidad, tendrá que adquirir otra impresora que no sea Sinclair y conectarla al ordenador por medio del interface RS232 en el Interface 1 ZX.

La impresora se utiliza con LPRINT y LLIST (similares a PRINT y LIST) y COPY (que hace una copia de la imagen que aparece en la pantalla de TV).

Funciona imprimiendo una línea cada vez, de manera que, mientras dicha línea no esté terminada se almacena en una memoria de almacenamiento intermedio del Spectrum denominada *buffer de impresora*, que se comporta de forma similar a la última línea de la parte superior de la pantalla: cuando debe desplazarse la imagen con una operación scroll, se vacía el buffer y se imprime su contenido. Hay una pequeña diferencia, y es que se imprime el contenido del buffer cada vez que se utiliza LPRINT con un cambio de línea (newline) explícito o implícito, en lugar de esperar para averiguar si se imprime algo más. El contenido del buffer también se imprime si queda algo cuando el programa termina de manera normal (mensajes 0 ó 9).

TAB y ',' funcionan igual que en la TV, es decir, imprimiendo espacios. O sea, pueden utilizarse para imprimir el contenido del buffer.

AT—ver Glosario.

INK, PAPER, FLASH y BRIGHT no ejercen efecto alguno.

OVER e INVERSE funcionan como es de esperar.

CHR\$8 (retroceso) funciona, pero no produce ningún efecto en el extremo de la izquierda (ver 8.5).

8.5. Caracteres y teclas simbólicas

Los distintos tipos de caracteres son tratados de forma diferente por la presentación visual en la pantalla de TV. Los normales (códigos comprendidos entre 32 y 143) simplemente

imponen la correspondiente combinación de pixels en una posición de carácter, y aplican en ella la combinación temporal de colores. Los 21 caracteres gráficos definidos por el usuario (códigos 144-164) funcionan de igual forma, tomando la combinación de pixels que los define de la zona de la memoria reservada especialmente para ellos.

Los caracteres cuyos códigos se encuentran entre 165 (RND) y 255 (COPY) son 'simbólicos' y se descomponen en el conjunto de caracteres ordinarios que los constituyen, posiblemente con espacios situados antes o después. RND, INKEY\$, PI, <=, >= y <> no llevan nunca espacios, pero los demás van seguidos siempre por un espacio, y OR, AND, y los que van desde LINE hasta COPY van precedidos de un espacio, a menos que ya se haya imprimido previamente.

Los caracteres cuyos códigos están comprendidos entre 0 y 31 son *caracteres de control*, la mayoría de los cuales tienen varias aplicaciones en el sistema BASIC del Spectrum. Algunos de ellos están directamente relacionados con la instrucción PRINT, y otros deben completarse con uno o dos caracteres adicionales, que en tal caso pierden su significado normal y se utilizan para comunicar un número. Aunque todos ellos pueden escribirse directamente, como por ejemplo, 'PRINT CHR\$ 6;', generalmente existe un procedimiento mejor, en este caso, 'PRINT,'.

Código 6. Carácter para la impresión de una coma. PRINT, LPRINT e INPUT lo imprimen cuando hay una coma como separador. La pantalla de TV o la impresora producen el número necesario de espacios.

PRINT CHR\$ 6; equivale a PRINT,

Código 8. Retroceso. Hace que la posición de impresión retroceda un espacio sin afectar a ningún atributo. A causa de un fallo en la memoria ROM, si se intenta retroceder desde la posición AT 1,0 no ocurre absolutamente nada, y si se intenta el retroceso a partir de AT 0,0 se puede desbaratar la imagen de la pantalla.

Código 13. Newline (cambio de línea). PRINT CHR\$ 13; equivale a:

PRINT o PRINT'

Códigos 16-21. Caracteres de control INK, PAPER, FLASH, BRIGHT, INVERSE y OVER. Deben ir seguidos por un carácter suplementario para indicar el nuevo valor utilizado en la combinación temporal de colores. Por ejemplo:

PRINT CHR\$ 18; CHR\$ v; equivale a PRINT FLASH v;

La mayor parte de ellos pueden teclearse directamente (véase el apartado 3.3).

Código 22. Control AT, seguido por dos caracteres que indican el número de línea y el de columna. PRINT AT 1,c; es equivalente a PRINT CHR\$ 22; CHR\$ 1; CHR\$ c;

Código 23. Control TAB, seguido por dos caracteres que indican el número de columna con dos bytes, primero el menos significativo. Si c es igual o menor que 255, entonces PRINT TAB c; equivale a PRINT CHR\$ 23; CHR\$ c; CHR\$ 0;

Los demás códigos de control no son enviados normalmente por PRINT ni son reconocidos por la IV, que los representará con ? (salvo CHR\$ 9, que se ignora).

9. Almacenamiento en cinta cassette

Existen cuatro sentencias relacionadas con la grabación en cassette:

SAVE almacena información del ordenador en la cinta.

VERIFY vuelve a leer la información almacenada para comprobar que es correcta.

LOAD carga información tomada de una cinta en el ordenador.

MERGE combina un programa tomado de una cinta con un programa que ya está en el ordenador.

El formato exacto de las sentencias depende del tipo de información que se almacena: puede tratarse de un programa en BASIC, una matriz (de números o de caracteres) o un bloque de bytes tomados de la memoria. Estudiaremos por separado cada uno de estos tres tipos.

Todo lo que se almacene en una cassette recibe un *nombre de archivo*, una cadena que puede contener entre 1 y 10 caracteres.

VERIFY, LOAD y MERGE también contienen un nombre de archivo, que puede ser uno concreto (sólo se usan los 10 primeros caracteres), o dejarse en blanco, en cuyo caso se tomará el primer archivo apropiado que se encuentre.

Los nombres de archivo se diferencian de los de variables en que sí se diferencian las letras mayúsculas de las minúsculas. Las tres sentencias mencionadas en el párrafo anterior aceptarán únicamente un archivo que contenga el tipo adecuado de información, así que no importa que dos archivos de distinta clase tengan el mismo nombre.

Los errores más comunes son:

4 (LOAD, MERGE): No hay espacio de memoria suficiente para el archivo de programa o matriz. No se pierde la matriz ni el programa antiguo. Compruebe que no está intentando cargar un programa escrito para 48K en un Spectrum de 16K.

F (SAVE): El nombre de archivo se ha dejado en blanco o es demasiado largo.

R (VERIFY, LOAD, MERGE): hay varias posibilidades:

1. Al Spectrum no le gusta el archivo. Puede ser que la cinta esté estropeada, pero inténtelo de nuevo.

2. (VERIFY): no se ha aprobado el examen de verificación. Inténtelo de nuevo. Si tampoco sale bien, repita la grabación (SAVE).

3. (Archivo de bytes con VERIFY o LOAD): El archivo es demasiado largo. No se habrán leído los bytes. (Véase el apartado 9.3).

9.1. Programas

Cuando se graba un programa, las variables se graban junto con él. Hay tres modalidades de grabación:

SAVE nombre de archivo

SAVE nombre de archivo LINE

SAVE nombre de archivo LINE número de línea

(‘número de línea’ es una expresión entera). Por ejemplo:

SAVE “fruteros” LINE 200

Los formatos que incluyen la palabra **LINE** graban el programa en forma de 'cargar y ejecutar', es decir, nada más cargar el programa con **LOAD** (pero no con **MERGE**), comenzará automáticamente su ejecución por la línea indicada, o desde el principio si se omite el número de línea.

LOAD (**VERIFY** y **MERGE** son similares) adopta la forma **LOAD** nombre de archivo

LOAD borra el programa antiguo y las variables antes de cargar el nuevo, pero **MERGE** produce una combinación de los dos, desechando una línea o variable antigua sólo en el caso de que sea reemplazada por otra nueva que tenga el mismo número o nombre, respectivamente. **MERGE** nunca provoca automáticamente la ejecución.

9.2. Matrices

Un archivo de matriz almacena todos los números o caracteres que forman parte de una matriz. Las sentencias de control de grabación en cinta para los archivos de matriz son indicadas por **DATA** y un nombre de matriz seguido por dos paréntesis:

SAVE nombre de archivo **DATA** nombre de matriz ()
por ejemplo:

SAVE "direcciones" **DATA** a\$()

LOAD y **VERIFY** son similares, y **MERGE** no puede utilizarse.

LOAD crea una matriz nueva con el nombre especificado en la sentencia y tomando los elementos (números o caracteres) de la cinta. Borra cualquier matriz o cadena anterior que tuviera el mismo nombre.

9.3. Bytes

Los archivos de bytes contienen bloques de bytes tomados de la memoria y especificados indicando la dirección inicial *x* del bloque en la memoria y el número de bytes. Las sentencias de grabación se diferencian por el uso de **CODE** o **SCREEN\$**:

SAVE nombre de archivo **CODE** dirección inicial, longitud
donde 'dirección inicial' y 'longitud' son expresiones enteras, como por ejemplo:

SAVE "codigoZ80" **CODE** 60000, 5000

Con VERIFY o LOAD se pueden omitir algunas partes. En lo que respecta a LOAD, las posibilidades son:

LOAD nombre de archivo CODE

(carga los bytes en la zona de la memoria de donde se tomaron para la grabación)

LOAD nombre de archivo CODE dirección inicial

(permite cargar el bloque en una zona distinta de la memoria)

LOAD nombre de archivo CODE dirección inicial, longitud (produce error R si el archivo de bytes supera la longitud especificada).

Las opciones con VERIFY son las mismas. MERGE no es aplicable.

SCREEN\$ es, precisamente, una forma abreviada de decir 'CODE 16384,6912' en sentencias SAVE, VERIFY y LOAD, puesto que define la zona de la memoria que contiene el archivo de imagen; por ejemplo:

SAVE "imagen" SCREEN\$

10. Entrada y salida en general

Esta sección está dedicada, sobre todo, al Interface 1 ZX, y a la forma en que los dispositivos periféricos tales como el Microdrive son controlados por un programa BASIC. Sin embargo, gran parte de los principios estudiados son aplicables a un ordenador Spectrum sin interface, aunque en forma restringida, como por ejemplo, en PRINT # 1 (véase el apartado 8.2).

La información transmitida puede ser de diversos tipos. Los programas, matrices y bytes utilizan SAVE *, VERIFY *, etc., y otro tipo de información —los *datos*— se envían con PRINT y LIST, y se leen utilizando INPUT e INKEY\$. La sentencia MOVE adicional es más general, y copia archivos de caracteres de un lugar a otro.

10.1. Canales

Esta terminología se refiere a la *transferencia de información* en ambos sentidos entre el programa del ordenador y otro elemento. Por ejemplo, en una sentencia PRINT, el programa produce caracteres que se envían a la pantalla de TV.

El 'otro elemento' al que nos hemos referido es un *dispositivo*, entendiéndolo como tal no sólo un aparato, sino también un método para trabajar en él con caracteres (así pues, la pantalla de TV posee *dos* dispositivos, los que hemos denominado 'parte superior' y 'parte inferior'). Sin embargo, el programa trabaja, en realidad, con un *canal*, que es algo más concreto. Utiliza un dispositivo dado, pero puede incluir información adicional, tal como un nombre de archivo en un cartucho de Microdrive.

En principio, un canal es bidireccional: el programa puede escribir en él (salida, como con PRINT) y también leer en él (entrada, como con INPUT). En la práctica, la mayoría de los canales dan error en una de las direcciones (error J, 'Reading a write file', lectura de un archivo de escritura, o 'Writing to a read file', escritura en un archivo de lectura).

Un canal se especifica por un carácter que representa el dispositivo (sin distinguir entre mayúsculas y minúsculas), posiblemente seguido por un punto y coma (;) y cualquier otro dato necesario: número de unidad de discos, nombre de archivo, etcétera. Sin el Interface 1 ZX, los canales son

"s"—parte superior de la pantalla (sólo escritura)

"p"—impresora ZX (sólo escritura)

"k"—teclado y parte inferior de la pantalla (bidireccional)

Con el Interface, existen también los dispositivos "m" (Microdrive), "t" y "b" (RS232), y "n" (red), explicados en la Sección 11. Como ejemplos de indicativos de canales que los emplean podemos citar

"m";3;"animales"

y

"n";53

Antes de poder utilizar un canal, debe ser *abierto* para que el programa lo sepa, y después debe ser *cerrado* de nuevo. Algunas sentencias lo hacen automáticamente, pero si no, debe ocuparse de ello el usuario empleando OPEN (abrir) y CLOSE (cerrar). Es muy fácil olvidarse de cerrar archivos, pero puede ser

importante. Los tres canales convencionales siempre están abiertos.

Errores comunes:

4. Muchos canales ocupan por completo la memoria cuando se abren.

F o 'Invalid device expression' (expresión inválida de dispositivo): en el indicativo del canal no se ha reconocido el carácter correspondiente al dispositivo.

10.2. Corrientes

Las corrientes son las salidas del programa que están conectadas a los canales. Existen 16 corrientes, llamadas #0, #1, #2, ..., y #15. Se utilizan, por lo general, en sentencias PRINT e INPUT, como por ejemplo, PRINT #6;"Direccion". Para estas dos sentencias y algunas otras, es necesario abrir los canales y conectarlos a corrientes (utilizando para ello OPEN) antes de poderlas utilizar. No obstante, con algunas sentencias (tales como SAVE * y LOAD *) en las que la entrada o la salida se verifica toda de una vez, no se emplean corrientes. Estas sentencias abren un canal automáticamente, cumplen su cometido, y vuelven a cerrar el canal.

Las corrientes #0, #1, #2 y #3 son las corrientes *estándar* utilizadas por el sistema BASIC, y están conectadas a canales convencionales.

#0, conectada al canal "k", es la corriente a través de la cual entrar en el ordenador los comandos y las líneas de programa que se han tecleado.

#1, con el canal "k", es la corriente normalmente utilizada por INPUT; en otras palabras, INPUT utiliza la corriente #1 a menos que un elemento # indique otra cosa (véase el apartado 10.4).

#2, con el canal "s", es la corriente normalmente empleada por PRINT y LIST (y también por CAT).

#3, con el canal "p", es la corriente empleada normalmente por LPRINT y LLIST.

Aunque, en principio, estas corrientes se pueden conectar a otros canales diferentes, tal procedimiento es desaconsejado por Sinclair, y parece ser que a causa de ciertos fallos del sistema, no es de fiar en determinadas circunstancias. Algunos ejemplos en los que podría ser de utilidad son:

OPEN #3;"t" para que LPRINT utilice el Interface RS232

OPEN #0;"n";2 para aceptar comandos de otro Spectrum

Errores.—0 si se intenta usar una corriente que no se ha abierto, e 'Invalid stream number' (número inválido de corriente) si el número no está entre 0 y 15.

10.3. Caracteres

La comunicación entre el canal y el programa se verifica con los 256 caracteres del Spectrum, pero el canal tiene libertad para tratarlos de cualquier manera antes de enviarlos. Este hecho es más claro con los caracteres de control (del 0 al 31) y con los simbólicos (del 165 al 255). El Microdrive, por ejemplo, transmite y almacena dichos caracteres sin alterarlos, mientras que la pantalla de TV descompone los caracteres simbólicos (los deletrea) y actúa en respuesta a los caracteres de control (ver apartado 8.5). El interface RS232 tiene dos canales, uno de los cuales transforma los caracteres de manera muy parecida a como lo hace la TV, pero no así el otro.

Recuerde que en las sentencias PRINT, las funciones TAB y AT, los elementos de color, la coma separadora y el cambio de línea newline, funcionan enviando caracteres de control al canal que se está utilizando; corresponde al canal interpretarlos según venga al caso (véase el apartado 8.5).

En la entrada, muchos caracteres de control poseen significados concretos que afectan al buffer de entrada (memoria intermedia donde se conservan los caracteres hasta que llegue la instrucción ENTER, y con ella, el momento de interpretarlos). El buffer de entrada funciona, en general, del mismo modo que cuando se está tecleando, aunque no se manifieste en la parte inferior de la pantalla. Observe la siguiente lista.

Código 6 (separador ','): se inserta en el buffer (en realidad es un fallo, debería funcionar como CHR\$ 14).

Códigos 7-13: como EDIT, ⇐, ⇒, ⇅, ⇆, DELETE y ENTER cuando se está tecleando (observe que CHR\$ 13 es 'newline' en la salida y ENTER en la entrada).

Código 14: para comandos e INPUT funciona como ENTER, mientras que para INPUT LINE se inserta en el buffer.

Código 15: se inserta en el buffer.

Códigos 16-21 (controles de color): se insertan en el buffer junto con el carácter suplementario que va a continuación.

Códigos 22 y 23 (AT, TAB): para comandos e INPUT ignora dos caracteres suplementarios y luego se comporta como ENTER. Para INPUT LINE, se inserta en el buffer con dos caracteres suplementarios.

Todos los demás caracteres son simplemente insertados en el buffer.

10.4. PRINT, LPRINT, INPUT, LIST, LLIST, INKEY\$

Todos ellos funcionan por medio de corrientes. Utilizan su corriente convencional siempre que un elemento # no indique otra cosa. Por ejemplo:

```
PRINT "mensaje a la TV"; #5;23 *2
```

envía el "mensaje a la TV" a la corriente #2 (que es la convencional de PRINT), y "46" seguida por un newline CHR\$ 13 (para el final de la sentencia) a la corriente #5. Si viene a continuación otra sentencia PRINT, volverá a empezar utilizando la corriente #2 (y no la #5).

LIST es similar, pero más sencilla si cabe porque hay un elemento # a la derecha, justo al principio, o no lo hay. Es decir,

```
LIST #7
```

```
LIST #8;100 (o LIST #8,100)
```

LPRINT y LLIST son idénticos a PRINT y LIST, con la salvedad de que tienen la corriente #3 como convencional, en lugar de la #2. COPY no usa corriente, y sólo se aplica con la impresora ZX.

INPUT tiene como corriente convencional la #1. Si se requiere INPUT para una variable, los caracteres se extraen de la corriente y se almacenan en un buffer especial de entrada (véase también el apartado 10.3; este buffer de entrada está separado de cualquier buffer establecido por el canal) hasta que se reciba un carácter que marque el final (generalmente CHR\$ 13 por ENTER). Entonces, se interpretan los caracteres como una expresión (error C si se ha cometido un fallo de sintaxis, excepto con el canal "k").

INPUT sirve tanto para leer (variables) como para escribir (mensajes), y su canal estándar, "k", es bidireccional. La mayoría no lo son, y por ello es importante no intentar producir mensajes con ellos ni utilizar comas como separadores (pues envían CHR\$ 6). O sea, se escribe INPUT #6;x;y en lugar de INPUT #6,x,y.

Con canales que no sean "k", los caracteres de entrada suelen proceder de una sentencia PRINT, ya sea directamente (es decir, por la red) o indirectamente (por medio de un archivo del Microdrive). La salida de PRINT, en consecuencia, debe ser aceptable por parte de INPUT. Tenga cuidado con las dos cuestiones siguientes.

Como INPUT espera ENTER al final de la entrada, PRINT debe aportar cambios de línea —las dos cosas son CHR\$ 13—. O sea, la salida de

```
PRINT #6;10*15
```

(con dos cambios de línea uno de los cuales es explícito y otro implícito) es adecuada para ser leída luego por

```
INPUT #7;m;n
```

pero PRINT #6;10,15, daría error C porque la salida contiene comas como separadores (CHR\$ 6) donde INPUT requiere ENTER.

La segunda cuestión tiene que ver con las cadenas que contienen en su interior el signo de comillas ". Una entrada INPUT normal espera que las comillas se dupliquen, pero PRINT no lo habrá hecho. Por lo tanto, es más seguro —a menos que se decida utilizar todas las posibilidades de las expresiones alfanuméricas generales— recurrir a la forma INPUT LINE.

INKEY\$ se utiliza para formar un único carácter de una corriente.

Las corrientes pueden dar lugar a resultados extraños con las combinaciones de colores. Cuando se selecciona uno de los canales convencionales, entra en juego la combinación permanente de colores apropiada. Así pues, el efecto de un elemento de color no se extiende hasta después de un #0, 1, 2 ó 3.

A causa de un fallo, si se activa un canal de periférico y más adelante se emplea una sentencia de color, los controles de color serán enviados al canal equivocadamente.

El remedio consiste en utilizar 'PRINT;' antes de la sentencia de color:

```
10 PRINT #6;"algo"  
20 PRINT;  
30 FLASH 0
```

Se produce el error 8 cuando se intenta leer un canal en el que no quedan caracteres. Esto es posible únicamente con el Microdrive (al final de un archivo) y con la red (cuando Iris —ver 11.2— ha cerrado su canal); en otro caso, INPUT esperará.

10.5. SAVE *, VERIFY *, LOAD *, MERGE *

Observe las estrellas necesarias en estas sentencias de Interface 1 ZX. Su papel es bastante parecido al de los canales en las sentencias correspondientes de utilización de cassette. No utilizan corrientes, sino que abren un canal, realizan la operación, y cierran el canal. El nombre de archivo normalmente empleado con las cassettes se sustituye por un indicativo o especificador de canal. Por ejemplo:

```
SAVE * "m";2;"programa" LINE 50      (Microdrive)  
VERIFY * "b" DATA a$()              (RS232)
```

Son válidas las mismas opciones que para la grabación en cinta, pero no obstante, los programas grabados utilizando LINE no pueden volver a cargarse con MERGE *, lo cual los protege de ser inspeccionados.

Errores: 4 (LOAD *, MERGE *): Ver Secciones 9 y 10.1.

'Code error' (MERGE *): Cuando se utiliza MERGE * con un archivo de matriz o de bytes, o con un programa de 'cargar y ejecutar'.

'Verification has failed' (ha fallado el examen de verificación) (VERIFY *): error R (10.1).

'Wrong file type' (tipo de archivo incorrecto) (VERIFY *, LOAD *): el tipo de archivo —de programa, matriz o bytes— no es el que se requiere; también, con VERIFY *, LOAD * y MERGE *, el archivo es un *archivo de datos*, es decir, ha sido establecido por PRINT etc., y no con SAVE *.

10.6. OPEN, CLOSE, MOVE, ERASE, CAT, FORMAT

Estas instrucciones no se utilizan normalmente, salvo con el Interface 1 ZX, sin el que únicamente OPEN y CLOSE tienen alguna aplicación. Los demás pueden ser aceptados en programas (frecuentemente con sintaxis equivocada) pero dan lugar al error 0. Véanse las Secciones 4 y 11 para más detalles.

OPEN abre un canal y lo conecta a una corriente.

CLOSE cierra un canal, desconectándolo de la corriente. Esto suele ser una buena idea cuando se ha terminado de trabajar con la corriente, y muchas veces es incluso imprescindible.

MOVE copia caracteres de un lugar a otro, utilizando corrientes (ya abiertas previamente) o canales (abiertos automáticamente).

ERASE borra un archivo de un cartucho de Microdrive.

CAT produce una lista de los archivos existentes en un cartucho de Microdrive.

FORMAT tiene diversas aplicaciones, dependiendo del dispositivo con el que se emplee. Normalmente se trata de algún tipo de inicialización.

11. El Interface 1 ZX

Este interface contiene un controlador para el Microdrive ZX (aunque no el propio Microdrive —se pueden utilizar hasta 8

de ellos separadamente— ni sus cartuchos), conexiones para Red Local de ordenadores Spectrum, un interface RS232, y una unidad adicional ROM de 8K con BASIC para manejar todo esto. En la Sección 10 se explican los aspectos generales que se aplican a cada uno de estos tres dispositivos, mientras que en esta sección nos dedicaremos a estudiar la forma de abrir los canales, el tratamiento de los caracteres por los canales, y el funcionamiento de las diversas sentencias.

11.1. El Microdrive ZX

El Interface 1 ZX puede controlar hasta 8 unidades Microdrive, numeradas desde el 1 (la más próxima al Interface) hasta el 8 (la más alejada). Cada Microdrive puede contener un cartucho, igual que una grabadora puede contener una cinta cassette. Antes de poder emplear un cartucho, debe ajustarse su formato (ver FORMAT, más adelante) una vez, operación en la que también se le asigna un nombre.

No conviene confiar en un solo cartucho. Cuantas más veces se utilicen, menos de fiar será la información que contengan, de manera que es fundamental conservar copias de seguridad de los que guarden algo importante.

La información se conserva en un cartucho en forma de archivos, cada uno de ellos identificado por un nombre compuesto por un número máximo de 10 caracteres. Un cartucho puede contener unos 180 archivos si son pequeños (de 512 bytes o menores), y no tantos si se trata de archivos grandes.

Un canal de Microdrive es un archivo único, y su indicativo es “m”; *número de unidad Microdrive*; *nombre del archivo* (‘número de unidad Microdrive’ es una expresión entera, y ‘nombre de archivo’ es una expresión alfanumérica).

Ejemplos:

OPEN #5;“m”;1;“setas”

SAVE * “m”;2;“shadowROM” CODE 0,8192

El canal se abre para lectura (si ya existía previamente) o para escritura (en caso contrario). Nunca es bidireccional.

Recuerde la posibilidad de autoejecución (consulte RUN en el Glosario).

OPEN # corriente; "m"; unidad Microdrive; nombre de archivo
'corriente' y 'unidad Microdrive' son expresiones enteras,
'nombre de archivo' es una cadena. Se produce el error
'Wrong file type' (tipo incorrecto de archivo) si el archivo fue
creado originalmente por medio de SAVE *.

CLOSE # corriente

Es esencial cerrar un archivo de Microdrive cuando se haya
terminado de escribirlo, pues de lo contrario no podrá ser
leído posteriormente.

MOVE origen TO destino

Se produce el error 'Wrong file type' (tipo incorrecto de
archivo) si el origen es un archivo de Microdrive creado
originalmente con SAVE *.

ERASE "m"; unidad Microdrive; nombre de archivo

Se borra el archivo indicado, dejando libre el espacio que
ocupaba en el cartucho. Puede utilizarse también para borrar
un archivo que, por accidente, no fue cerrado.

CAT unidad Microdrive

CAT # corriente; unidad Microdrive

Produce un catálogo del cartucho que haya en la unidad
Microdrive especificada, que consta del nombre del cartucho,
una lista de los nombres de los archivos, y del espacio libre
que queda, expresado en Kilobytes. El catálogo suele
escribirse en la pantalla (#2), pero también es posible enviarlo
por cualquier otra corriente, según se indique. Si el nombre
de un archivo empieza por CHR\$ 0, se trata de un *archivo*
secreto, y no aparece en el catálogo generado por CAT.

FORMAT "m"; unidad Microdrive; nombre del cartucho

Piénselo dos veces antes de utilizar esta sentencia. 'nombre
del cartucho' es una expresión alfanumérica que contiene de
1 a 10 caracteres (en caso contrario se incurre en el error
'Invalid name' —nombre inválido—). Sirve para determinar
el formato de un cartucho, dándole el nombre indicado y
borrando los archivos que hubiera en él. Esta operación debe
realizarse siempre con un cartucho en blanco antes de poderlo
utilizar, pero también sirve para borrar todos los archivos de
un cartucho que ya estuviera en servicio. Tenga mucho

cuidado con **FORMAT**, porque si al utilizarlo se equivoca de cartucho o de unidad Microdrive, se arrepentirá con toda seguridad.

Errores comunes:

'Device write protected' (dispositivo protegido frente a escritura): Cuando se intenta escribir, borrar (**ERASE**) o ajustar el formato (**FORMAT**) con un cartucho al que se le ha quitado la lengüeta de protección.

'File not found' (no se encuentra el archivo): Cuando se utiliza **VERIFY ***, **LOAD *** o **MERGE *** con un archivo que no existe, o cuando se intenta leer un archivo sin cerrar (porque se le haya olvidado cerrarlo con **CLOSE**, porque algo haya salido mal cuando se estaba escribiendo el archivo, o quizás porque se haya deteriorado el cartucho).

'Invalid drive number' (número inválido de unidad Microdrive) Cuando el número no está comprendido entre 1 y 8.

'Invalid name' (nombre inválido): el nombre de archivo o cartucho no tiene ningún carácter o tiene más de 10.

'Microdrive full' (Microdrive lleno): No queda espacio suficiente para lo que se está intentando escribir (que no será cerrado y debe borrarse con **ERASE**).

'Microdrive not present' (no hay Microdrive): No hay Microdrive, o no hay cartucho, o no se ha determinado el formato del cartucho.

11.2. Circuito local (red)

Pueden interconectarse hasta 64 ordenadores Spectrum, cada uno con su propio Interface 1 ZX, formando una red. Cada uno utilizará **FORMAT** para atribuirse un *número de estación* entre 1 y 64 (preferiblemente distinto de los demás).

Normalmente, los mensajes que se transmiten entre dos estaciones son privados, es decir, sólo incumben a la estación emisora y a la receptora, pero también es posible emitir o 'difundir' *mensajes públicos*, a disposición de la estación que esté interesada en recibirlos. Para los mensajes privados, el Spectrum

del otro extremo se denomina tradicionalmente 'Iris' (palabra griega que significa lo mismo que 'Spectrum' en latín); así pues, o bien se está enviando un mensaje a 'Iris' o bien se está recibiendo un mensaje de 'Iris'.

Tanto su estación como Iris deben abrir canales de red. Su indicativo de canal es "n"; número de estación de Iris. Ejemplos:

```
OPEN # 10; "n"; 15  
LOAD * "n"; 29 CODE
```

Un canal de red se utiliza para escritura (de su estación a Iris) o para lectura (de Iris a su estación), pero no para lectura y escritura al mismo tiempo. La aplicación depende de lo primero que se haga con el canal, una lectura o una escritura.

Cuando se escribe un mensaje, será enviado repetidamente hasta que lo lea Iris, y sólo entonces su programa seguirá adelante. Similarmente, si Vd. desea leer un mensaje, su programa aguardará hasta que Iris envíe algo (con INKEY\$ # sucede otra cosa: produce una cadena vacía "" si el canal que va de Iris a su estación sigue abierto pero no se está recibiendo nada en el momento).

Para la difusión general, tanto la estación emisora como las receptoras utilizan el canal "n";0 y el programa del emisor nunca espera a que los receptores hayan recibido el mensaje; sin embargo, la emisión tiene lugar en este caso muy lentamente para que todas las estaciones receptoras tengan la oportunidad de ocuparse de ella. INKEY\$ # en un canal de difusión siempre espera a la llegada de un mensaje.

OPEN # corriente; "n"; número de estación

CLOSE # corriente

Después de escribir un mensaje en una corriente de la red, es esencial completar la transmisión. Pone en orden el sistema después de una lectura, y también es necesario si se pretende volver a utilizar la corriente para otra escritura.

MOVE origen TO destino

FORMAT "n"; número de estación

Esta sentencia determina su propio número de estación (entre 1 y 64, so pena de cometer el error 'Invalid station

number'). Las estaciones Iris que estuvieran comunicándose con Vd. previamente pueden seguir utilizando su antiguo número de estación. Cuando empiece a utilizar el Interface 1 ZX después de activar el equipo, su estación recibe automáticamente el número 1. PEEK 23749 le dirá el número que tiene su estación.

El error más común es 'Invalid station number' (número inválido de estación): debe estar comprendido entre 1 y 64, siendo 0 para los canales de difusión general.

11.3. Interface RS232

El interface RS232 funciona con todas las velocidades de transmisión (expresadas en baudios) convencionales, las cuales son determinadas por FORMAT, con 8 bits de datos, sin bit de paridad y un bit de parada. Tiene dos líneas de datos (emisión y recepción), dos líneas de 'saludo de inicio de comunicación' (intercambio de indicativos y señales de control: CTS para indicar que el Spectrum está listo para recibir, y DTR para que los otros dispositivos indiquen que están preparados para recibir), y otras dos líneas de 0V y 9V.

Hay dos canales que utilizan el interface RS232, especificados con "b" (binario) y "t" (texto), tal como en

OPEN # 9; "b"

Los dos son bidireccionales, y ambos funcionan con la misma velocidad de transmisión (baudios). Mientras que "b" envía los caracteres sin modificar, "t" tiene que procesar cualquier carácter que sea exclusivo del Spectrum para que los textos producidos por este ordenador puedan imprimirse en otras impresoras. Expande (deletrea) los caracteres simbólicos ('tokens'), de códigos comprendidos entre 165 y 255, incluyendo en ocasiones espacios superfluos; sustituye los caracteres gráficos (códigos 128-164) por '?', y suprime todos los caracteres de control (códigos 0-31), excepto 'newline' (13), que se envía como 13 seguido por 10 (códigos que en ASCII corresponden, respectivamente, al retorno de carro y cambio de línea). Por desgracia, este funcionamiento de "t" es bastante improvisado. Con LIST no hay problemas, pero no así con PRINT, porque no utiliza la coma como separador; AND y TAB son controlados correctamente.

En la lectura, "t" lee el octavo bit dato como 0, independientemente del valor que tenga, lo cual resulta útil para eliminar los bits de paridad. Por su parte, "b" lee fielmente los ocho bits.

Para archivos de datos se puede emplear cualquier canal; SAVE *, VERIFY *, LOAD * y MERGE * deben utilizar el "b".

INKEY\$ produce el resultado nulo "" si no hay ningún carácter presente en el momento.

OPEN # corriente; "b"

OPEN # corriente; "t"

Es posible abrir varias corrientes para el mismo canal del RS232.

CLOSE # corriente

Esto es conveniente, pero no imprescindible. Produce un ahorro de 11 bytes de memoria.

MOVE origen TO destino

Los canales del RS232 no indican nunca el final de un archivo, de manera que si se utiliza uno como origen, la sentencia MOVE no acabará nunca, si no es con BREAK.

FORMAT "b"; velocidad de transmisión (baudios)

FORMAT "t"; velocidad de transmisión (baudios)

'velocidad de transmisión' es una expresión entera cuyos posibles valores son: 50, 110, 300, 600, 1200, 2400, 4800, 9600 y 19200. Con esta sentencia se determina la velocidad de transmisión, expresada en baudios, del RS232. Si se indica una de las velocidades no reconocidas, se utilizará la inmediatamente superior. No importa el canal que se emplee, pues los dos tienen la misma velocidad de transmisión.

12. Mensajes informativos

En esta sección facilitamos algunos detalles que no aparecen en las listas de mensajes de los manuales del Spectrum y el Interface 1 ZX. Si un mensaje de error le ha confundido, el

primer paso es descubrir la sentencia o función causante del error. Entonces habrá una explicación del motivo en una de las tres listas, y si se menciona únicamente la palabra clave, sin ninguna explicación, buscarla en el Glosario, Sección 4.

- 3 Subscript error (error de subíndice)—ver 5.4, 5.5
- 4 Out of memory (falta memoria)
LET, INPUT, READ, FOR, DIM —no queda espacio en la memoria para la nueva variable o matriz.
LOAD, MERGE —ver 9 (10.5 para LOAD * y MERGE *).
OPEN (o como quiera que se haya abierto un canal) —ver 10.1.
GO SUB, FN —ver 6.4 y 5.8 respectivamente.
También se produce cuando no queda espacio para poder evaluar una expresión —posiblemente con operadores de cadenas, tales como +.
- 5 Out of screen (Falta espacio de pantalla)—INPUT o PRINT # 1 (ver 8.2); AT.
- 6 Number too big (Número demasiado grande)—ver 5.1; también TAN y EXP.
- 8 End of file (Fin de archivo)—ver 10.4; también INKEY\$ #.
- A Invalid argument (Argumento inválido)—posiblemente con ↑ (si el número de la izquierda es negativo).
- B Integer out of range (Número entero fuera del margen aceptable)
LIST, LLIST, RESTORE, DIM, PEEK, USR (argumento numérico), IN, CLEAR, RANDOMIZE, PAUSE, TAB, subíndices (con matrices o fragmentos de cadenas), SAVE, VERIFY o LOAD (con CODE) —un número está fuera del margen limitado por 0 y 65535.
CHR\$, BORDER, INK, PAPER, FLASH, BRIGHT, OVER, INVERSE, #, OPEN #, CLOSE # —número no comprendido entre 0 y 255.
RUN, GO TO, GO SUB —el número de línea no está comprendido entre 0 y 61439.
AT, SCREEN\$, ATTR, PLOT, POINT, DRAW, CIRCLE, POKE, OUT, BEEP, INKEY #.
- C Nonsense in BASIC (no tiene sentido en BASIC)—VAL, VAL\$
INPUT error de sintaxis en la entrada.

Puede producirse en lugar del error F si está conectado el Interface 1 ZX.

- D BREAK—CONT repeats (se repite BREAK)
Funcionamiento con cassette, Impresora ZX o con la red, y entrada por RS232 —se ha pulsado SPACE.
Funcionamiento con Microdrive o salida por RS232 —se ha pulsado CAPS SHIFT y SPACE.
Después de 'scroll?' —se ha pulsado N, SPACE o STOP.
- F Invalid file name (nombre de archivo inválido) —SAVE (ver 9); ver también 10.1.
- J Invalid I/O device (dispositivo de entrada/salida no válido) —se ha intentado leer en los canales "s" o "p".
- L BREAK into program (hay un BREAK en el programa) —ver 3.6.
- N Statement lost (sentencia perdida)
RETURN, NEXT, CONTINUE —se ha intentado saltar a una sentencia que ha sido eliminada del programa en una edición, o a una parte de una línea de comando anterior.
RUN, GO TO, GO SUB —puede ocurrir cuando el número de línea está entre 32768 y 61439.
- O Invalid stream (corriente inválida)
MOVE, ERASE, CAT, FORMAT —si no está conectado el Interface 1 ZX.
#, CLOSE # —corriente sin abrir, o con un número igual o mayor que 16 (ver 10.2).
OPEN #, INKEY\$ #.

Los mensajes informativos del Interface 1 ZX carecen de números o letras de códigos. La mayoría son específicos del Microdrive, de la red o del interface RS232, y se explican en la Sección 11.

Invalid device expression (expresión de dispositivo inválida)
—Indicativos de canal (10.1); también OPEN.

File not found (archivo no encontrado) —Microdrive (11.1).

Program finished (programa terminado) —GO TO, RUN.

Reading a 'write' file (se ha intentado leer un archivo de escritura) —Microdrive o red (11.1, 11.2).

Stream already open (corriente ya abierta) —OPEN.

Writing to a 'read' file (se ha intentado escribir en un archivo de lectura) —Microdrive o red (11.1, 11.2).

Índice de materias

Almacenamiento en cinta, 64.

-- cassette, 64.

Archivo, 64, 68, 75.

- de bytes, 66, 67, 73.

- de datos, 67.

- de matrices, 66, 67, 73.

- de programas, 66, 67, 73.

Argumento, 2, 42, 47.

Atributo, 50.

Bidireccional, 68.

Bucle, 49.

Cadena, 36, 39.

Canal, 67.

Carácter, 62, 70.

- de control, 7, 62, 70.

- gráfico definido por el usuario,
56.

Color, 7, 50.

Comando, 9.

Combinación de colores, 52.

Cómo utilizar este Manual de Bol-
sillo, 1.

Condición, 43, 48.

Coordenadas, 51.

Corriente, 69.

Cursor, 5.

Dispositivo, 68.

Edición, 7, 10.

Entrada/Salida, 64, 67.

Error, 4, 82.

Estructura, 47.

Expresión, 1, 34.

Fragmentación, 39.

Función, 41, 46.

- definida por el usuario, 46.

Glosario, 11.

Gráficos, 50, 54, 56.

Impresora, 62.

Informe, 80.

Interface-1, 2, 67, 74.

-2, 3.

Iris, 78.

Línea multisentencia, 9, 47.

Listados, 11.

Matriz, 37, 40.

Microdrive, 67, 75.

Modo, 6.

Newline, 59, 72.

Número, 35.

Operador, 40, 42.

- de comparación, 42.

- lógico, 42.

Pantalla inferior, 4, 59.

- - y superior, 4, 56, 59.

- de TV, 4, 50, 56.

Pixel, 50.

Posición de carácter, 51.

Programa, 4, 10.

Red, 68, 77.
- local, 68, 77.
RS232, 68, 79.

Scroll (desplazamiento hacia arriba),
4, 57, 59.
Spectrum, 2.

Subrutina, 49.

Teclado, 6, 8.
Teclear, 3, 4, 8.
Token (símbolo), 62, 70.

Variable, 36.

OTROS TITULOS

Abatut, Aracil y Pun

Introducción a la automática integrada

Arroyo

Del Bit a la telemática. (Introducción a los ordenadores.)

Bishop

Estudio general del ordenador

Díaz, Lora-Tamayo y Pun

Diseño asistido por ordenador de circuitos electrónicos

Dworatschek

Introducción al proceso de datos

Friedrichs y Schaff

Microelectrónica y sociedad, para bien o para mal

Fry

Ordenadores

FUINCA

Bases de datos del mundo. Catálogo de los sistemas de información científica, tecnológica, social y económica accesibles desde España por medios telemáticos

Harrison

Proceso de datos

Martínez y Rieiro

66 programas en BASIC. De la EGB a la Universidad

Osgood y Molloy

Toma de decisiones en actividades comerciales (Ordenadores Apple)

Osgood y Molloy

Toma de decisiones en actividades comerciales (Ordenadores IBM)

Pozo

Curso de informática

Esta guía no pretende sustituir al manual de instrucciones del Sinclair-Spectrum, sino proporcionar una información fácilmente asequible sobre todas las cuestiones que el usuario pueda plantearse para obtener un mayor rendimiento de su ordenador personal. Redactado de forma sencilla y concisa, contiene una valiosa serie de datos sobre el manejo del Sinclair-Spectrum.



Alhambra
Ciencias y Técnica